A MILP model for N-dimensional allocation

Article *in* Computers & Chemical Engineering · December 2007

DOI: 10.1016/j.compchemeng.2007.02.006 · Source: DBLP

3 authors, including:

Tapio Westerlund
Åbo Akademi University
165 PUBLICATIONS **3,008** CITATIONS

SEE PROFILE

Available online at www.sciencedirect.com
ScienceDirect
Computers and Chemical Engineering 31 (2007) 1702–1714
Computers & Chemical Engineering
www.elsevier.com/locate/compchemeng

A MILP model for *N*-dimensional allocation

Joakim Westerlund [a], Lazaros G. Papageorgiou [b], Tapio Westerlund [a,*]

[a] *Process Design Laboratory, Faculty of Chemical Engineering, Åbo Akademi University, Turku 20500, Finland*
[b] *Centre for Process Systems Engineering, Department of Chemical Engineering, UCL (University College London), London WC1E 7JE, UK*

Received 24 March 2005; received in revised form 21 December 2006; accepted 5 February 2007
Available online 20 February 2007

Abstract

This paper presents a Mixed Integer Linear Programming (MILP) model for the solution of *N*-dimensional allocation problems. The applicability of the model is presented and demonstrated through some illustrative examples with different numbers of dimensions. Several problems, previously presented in the literature, are solved using the proposed model, such as, one-dimensional scheduling problems, two-dimensional cutting problems, as well as plant layout problems and three-dimensional packing problems. Additionally, some problems in four dimensions are presented and solved using the considered model. The presented model is applicable to a wide variety of allocation problems as it offers a general framework for modelling allocation problems with any given number of continuous or discrete dimensions. The presented problems are formulated as MILP problems where the first four dimensions usually are continuous spatial and time dimensions. Additional dimensions are often of a discrete nature.

Keywords: Optimisation; *N*-Dimensional allocation; MILP

1. Introduction and motivation

Allocation problems in different number of dimensions have been research topics of great relevance for many years. Most allocation problems are large-scale combinatorial optimisation problems, admitted in several different industrial applications. Numerous problem formulations for different kinds of allocation problems have been proposed during the last few decades. The block-layout design problem, originally formulated by Armour and Buffa (1963) was surveyed in Meller and Gau (1996) and also considered in Castillo, Westerlund, Emet, and Westerlund (2005). In Lodi, Martello, and Monaci (2002) some solution approaches for two-dimensional fixed orientation packing problems were investigated. Furthermore, the Process Plant Layout (PPL) problem has attracted much attention since heuristic solution approaches were presented in Gunn and Al-Asadi (1987), Suzuki, Fuchino, Muraki, and Hayakawa (1991) and Amorese, Cena, and Mustacchi (1991). A graph partitioning approach, for the PPL problem, was presented by Jayakumar and Reklaitis (1994) and an MINLP model was proposed by Penteado and Ciric (1996). MILP approaches for the Process Plant Layout problem were presented in Papageorgiou and Rotstein (1998) and in Patsiatzis and Papageorgiou (2002). Efficient solution approaches for the multi-floor PPL problem, based on iterative or decomposition strategies, were presented in Patsiatzis and Papageorgiou (2003) and in Jernström, Westerlund, and Papageorgiou (2004). A three-dimensional palletisation problem formulation was presented in Tsai, Malmstrom, and Hayakawa (1991) and a three-dimensional facility layout formulation in Barbosa-Póvoa, Mateus, and Novais (2002) and, additionally, a three-dimensional box packing formulation in Westerlund, Papageorgiou, and Westerlund (2005a). According to our knowledge, however, no general purpose model for allocation problems in any given number of dimensions has yet been published.

The *N*-dimensional allocation problem consists of items in *N*-dimensions. The items may be allocated inside a limited one-dimensional array, a two-dimensional area, a three-dimensional volume, and so forth. The array, area or volume is called *container* in all cases in order to simplify the terminology. For example, if a volume item, additionally, is connected with a time dimension determining its availability in time, this item should be allocated both in space and time. Nevertheless, such an item will still be allocated in an overall four-dimensional container using this terminology. To avoid peculiar problem-specific constraints, all items are assumed to be rectangular.

* Corresponding author. Tel.: +358 2 215 4458; fax: +358 2 215 4791.
E-mail address: twesterl@abo.fi (T. Westerlund).

0098-1354/$ – see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2007.02.006

Moreover, the problem formulation consists of an objective function and four main types of constraints; space constraints limiting the items to be allocated inside one of the containers, non-overlapping constraints preventing items from overlapping each other, orientation constraints allowing items to rotate in many dimensions, and container constraints defining if a certain container is used or not. The problem formulation determines the container allocation as well as the allocation of each item simultaneously.

Furthermore, allocation problems are large-scale combinatorial optimisation problems and the problem-size may grow rapidly with the number of dimensions. Hence, some formulation enhancement methods are presented, significantly cutting down the computational effort needed for the solution, thus, enabling the solution of larger problems without excluding the optimal solution. The formulation enhancement strategies are included in the formulation as additional constraints. Note, however, that the effectiveness of the formulation enhancement strategies greatly depends on the nature of the problem. Some strategies may work very well on a specific problem type but might not offer any significant benefits on another kind of problem.

2. Problem description

N-Dimensional allocation problems may be of a very different nature. Regardless of the number of dimensions, the main features are still the same. A number of items are to be allocated inside containers in one or many dimensions. The items do not necessarily represent physical objects; they might also, for example, represent time sequences in a production plan, as seen in Fig. 1.

Facility layout problems, strip-packing problems or single-floor Process Plant Layout problems, as illustrated in Fig. 2, are allocation problems in two continuous space dimensions. A few illustrative examples of problems in two dimensions are further presented in Section 4.2.

In Fig. 3, a three-dimensional packing problem with one container is illustrated. Different packing problems are further presented in Section 4.3. Some problems require additional discrete dimensions in addition to the continuous space dimensions to determine where a specific container is allocated in the discrete space.

In numerous packing problems discrete dimensions may be used, in addition to the continuous dimensions, in order to determine, for example, which containers are to be used. In Section 4.4 an allocation problem in four continuous dimensions is, furthermore, presented.

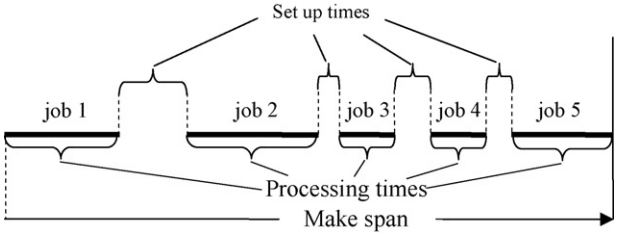

Fig. 1. A one-dimensional production schedule.

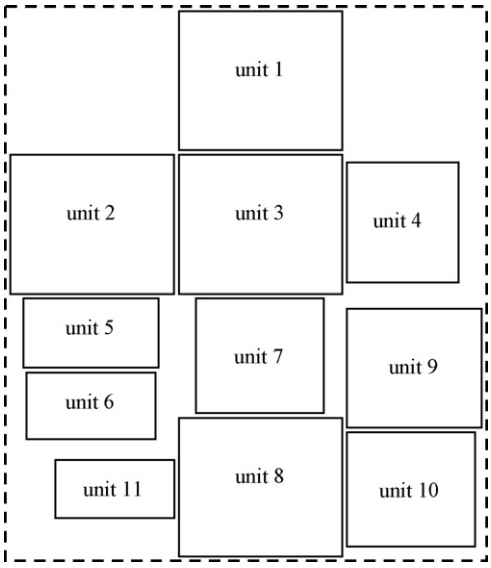

Fig. 2. Two-dimensional single-floor Process Plant Layout problem.

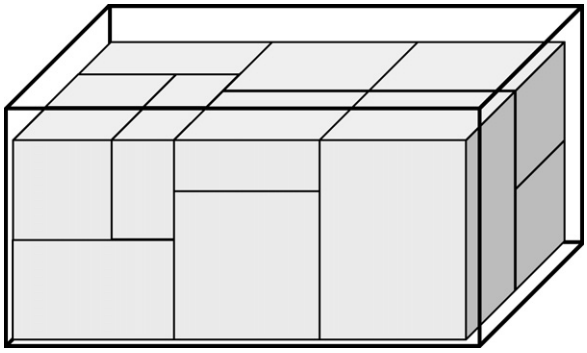

Fig. 3. Three-dimensional packing problem.

3. Problem formulation

The proposed mathematical formulation for *N*-dimensional allocation consists of an objective function subject to four main types of constraints, and some additional constraints. The four main constraints are: space constraints, overlap prevention constraints, orientation constraints and container constraints. The additional constraints may be included in the problem formulation in order to decrease the problem complexity and reduce the computational effort needed for solution.

3.1. Nomenclature

The model for the *N*-dimensional allocation problem is formulated by using vector notations in order to achieve a compact formulation. So, to write the expressions for an *N*-dimensional allocation problem in a compact form, we will first present the variables defining the coordinates, distances, sizes, and so forth. A rectangular item i in *N*-dimensions is represented by

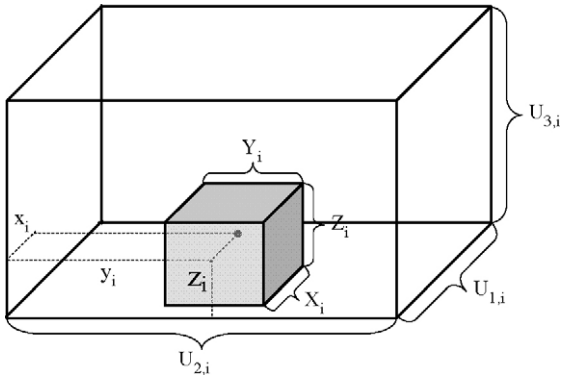

Fig. 4. Centroid coordinates and variable size for an item as well as the corresponding container in a three-dimensional problem.

N coordinates defining its centroid by $\mathbf{x}_i$ where $\mathbf{x}^T = (x, y, z, t, \ldots)$. The number of dimensions, N, is given by the dimension of the vector $\mathbf{x}$, $N = \dim(\mathbf{x})$. Consequently, the centroid coordinates of a rectangular box, i, in three dimensions is defined by $\mathbf{x}_i$, where $\mathbf{x}_i^T = (x_i, y_i, z_i)$, and the number of coordinates is $N = \dim(\mathbf{x}) = 3$. Furthermore, the size of a rectangular item i, in each of the N-dimensions, is defined by $\mathbf{X}_i$ where $\mathbf{X}_i^T = (X_i, Y_i, Z_i, \ldots)$. X_i, Y_i and Z_i are the length, width and height of the item. Observe, nonetheless, that the elements of $\mathbf{X}_i$ are variables, and not given *a priori*, since an item is allowed to rotate in a certain number of dimensions. The dimension of $\mathbf{X}$ is equal to the dimension of $\mathbf{x}$, that is, if a rectangular item is defined by N coordinates it is also defined in size in all these directions. Furthermore, $\mathbf{v}$ is defined by $\mathbf{v}^T = (l, w, h, \ldots)$ where $\mathbf{v}$ is a vector defining the size of each item, regardless of orientation. The elements of the vector $\mathbf{X}_i$ will be selected from the elements in the vector $\mathbf{v}_i$.

The size of the container in which item i is located is defined by the vector $\mathbf{U}_i$. Observe that the elements in $\mathbf{U}_i$ are variables defining the size of the container. The number of containers is K, and each container has a given size defined by the vector $\mathbf{V}_k$. $\mathbf{V}_k^T = (L_k, W_k, H_k, \ldots)$ where L_k, W_k, H_k, ... are the given length, width, and height of the container k. The elements of a vector, $\mathbf{U}_i$, will be selected from the elements in the vectors $\mathbf{V}_k$. The centroid and the variable size for an item in three dimensions and the container where it is located are illustrated in Fig. 4.

3.2. The objective function

The objective function considers the minimisation of the total cost of the allocation pattern, and may include numerous cost factors to suit each specific problem. The most uncomplicated objective function is simply a sum of container costs connected to each container as shown in the following equation:

$$\min\left(\sum_{k=1}^{K} C_k \cdot D_k\right) \tag{1}$$

In Eq. (1), C_k stands for container costs and D_k binary variables defining the existence or non-existence of a container. Using the objective function in Eq. (1) the summarised container costs are minimised. In this case, however, the items are not enforced to be allocated as tightly as possible inside the containers. In some cases, the space occupied by items in some container direction may be worth minimising simultaneously with the container costs. In the presented model, optional additional costs may be included in the objective function as shown in Eq. (2). The additional costs in Eq. (2) make the allocation pattern tighter by pushing the items into a specific corner of the container. Different costs in each dimension may be given. Furthermore, in cases where items may be interconnected, the rectilinear distance between items located in the same container might be worth minimising. In objective function (2), the rectilinear distance, $\mathbf{d}_{ij}$, in every coordinate direction between each item is taken into account and multiplied by a specific cost vector, $\mathbf{c}_{ij}$, for each connection. The cost vector, $\mathbf{c}_{ij}$, may be used as a basis for the connection costs as well as the material handling costs between the considered items:

$$\min \sum_{k=1}^{K}(C_k \cdot D_k + \bar{\mathbf{C}}_k^T \cdot \mathbf{s}_k) + \sum_{j=2}^{J}\sum_{i=1}^{j-1}(\mathbf{c}_{ij}^T \cdot \mathbf{d}_{ij}) + \sum_{i=1}^{J}\bar{\mathbf{c}}_i^T \cdot \mathbf{x}_i \tag{2}$$

In Eq. (2), $\bar{\mathbf{C}}_k$ is a cost vector for the distances, $\mathbf{s}_k$, occupied by items in each coordinate direction for a container and $\bar{\mathbf{c}}_i$ is a cost vector connected to the centroid coordinate of each item. Constraints defining the distances, $\mathbf{s}_k$, are given in Eqs. (8a)–(8c). The rectilinear distances, $\mathbf{d}_{ij}$, between items i and j are defined in Eq. (9). Eqs. (9) and (8a)–(8c) are given in Section 3.4.1.

3.3. Main constraints

The four main constraints incorporated in the problem formulation are usually included in all problem types. In some cases, nonetheless, some of the main constraints might not be applied. This is the case, for example, when solving strip-packing problems in which the items are not allowed to rotate. The strip-packing problem is further presented in Section 4.2.1.

3.3.1. Space constraints

The space constraints used in the problem formulation are used to prevent items from being allocated outside the container perimeter. The size of each item, i, is specified in N-dimensions, the size being defined by $\mathbf{X}_i$ where $\mathbf{X}_i^T = (X_i, Y_i, Z_i, \ldots)$ and $i = 1, \ldots, J$ is the index of the actual item. J is the total number of items. The centroid of each item i is represented by $\mathbf{x}_i$, where $\mathbf{x}_i^T = (x_i, y_i, z_i, \ldots)$. Furthermore, $\mathbf{U}_i$, represents the variable size of a container in which item i is allocated, as defined in the following equation:

$$\mathbf{x}_i \geq \tfrac{1}{2}\mathbf{X}_i, \qquad \mathbf{x}_i \leq \mathbf{U}_i - \tfrac{1}{2}\mathbf{X}_i, \qquad \mathbf{U}_i = \sum_{k=1}^{K}\mathbf{V}_k \cdot \beta_{ki},$$

$$\sum_{k=1}^{K}\beta_{ki} = 1, \quad i = 1, 2, \ldots, J \tag{3}$$

$\mathbf{V}_k$ is a vector defining the given length, width, height, and so forth for container k according to $\mathbf{V}_k^T = (L_k, W_k, H_k, \ldots)$. In Eq.

(3), β_{ki} is a binary variable equal to 1 if item i is allocated in container k; 0 otherwise.

3.3.2. Overlap prevention constraints

The overlap prevention constraints are used to prohibit items from overlapping each other. Items may, nevertheless, overlap each other in $N-1$ dimensions but at least one dimension is to be left un-violated. Items in four dimensions, such as x, y, z and t, are accordingly, allowed to overlap each other, for example, in all space dimensions as long as the time dimension is not overlapped. Similar types of overlap prevention constraints for problems restricted to two or three dimensions are given in Patsiatzis and Papageorgiou (2002), Castillo and Westerlund (2005), Westerlund et al. (2005a) and Westerlund, Papageorgiou, and Westerlund (2005b). General overlap prevention constraints in N-dimensions are given in the following equations:

$$\frac{\mathbf{X}_i + \mathbf{X}_j}{2} + \boldsymbol{\delta}_{ij} \leq (\mathbf{x}_i - \mathbf{x}_j) + M \cdot (\mathbf{P}_{ij} + \mathbf{Q}_{ij}), \quad 1 \leq i < j \leq J \tag{4a}$$

$$\frac{\mathbf{X}_i + \mathbf{X}_j}{2} + \boldsymbol{\delta}_{ji} \leq (\mathbf{x}_j - \mathbf{x}_i) + M \cdot (\mathbf{P}_{ij} - \mathbf{Q}_{ij} + \mathbf{e}), \quad 1 \leq i < j \leq J \tag{4b}$$

$$\mathbf{P}_{ij} + \mathbf{Q}_{ij} \leq \mathbf{e}, \quad 1 \leq i < j \leq J \tag{4c}$$

$$\mathbf{e}^{\mathrm{T}}\mathbf{P}_{ij} + G_{ij} \leq N, \quad 1 \leq i < j \leq J \tag{4d}$$

$$\beta_{ki} + \beta_{kj} \leq G_{ij} + 1, \quad k = 1, 2, ..., K, \quad K \neq 1, \quad 1 \leq i < j \leq J \tag{4e}$$

$\mathbf{e}$ is a unit vector $\mathbf{e}^{\mathrm{T}} = (1, 1, 1, \ldots)$, and M an appropriate upper bound, significantly large to ensure non-overlap. In cases with several containers, G_{ij} is a binary variable equal to 1 if items i and j are allocated in the same container; 0 otherwise. If only one container is used, $G_{ij} = 1$ for all i and j. Observe that G_{ij} can be defined through a relaxed inequality when minimising Eqs. (1) and (2). $\mathbf{P}_{ij}$ and $\mathbf{Q}_{ij}$ are vectors of binary variables and $\boldsymbol{\delta}_{ij}$, and $\boldsymbol{\delta}_{ji}$ are vectors of distance parameters defining the minimum allowable distance, in all directions, between two specific items i and j.

If $G_{ij} = 0$, then constraints (4a) and (4b) are inactive. If, conversely, $G_{ij} = 1$, then constraints (4d) guarantee that at least one element (dimension) of the $\mathbf{P}_{ij}$ vector will have zero value. Note that constraints (4a) and (4b) are inactive for elements (dimensions) of $\mathbf{P}_{ij}$ with value equal to 1 irrelevant of the corresponding values of $\mathbf{Q}_{ij}$. Therefore, only the $\mathbf{P}_{ij}$ element with zero value will safeguard non-overlapping between i and j. If then the corresponding $\mathbf{Q}_{ij}$ value is zero, constraints (4b) are inactive while constraints (4a) force i to be "after" j. Similarly, if the $\mathbf{Q}_{ij}$ element is equal to one then constraints (4a) are inactive and constraints (4b) guarantee that i is "before" j. Detailed proof that the non-overlapping constraints above do hold in N-dimensions is provided in Appendix A.

The $\boldsymbol{\delta}_{ij}$ may be used, for example, in two-dimensional layout problems for safety reasons or if a service passage is needed between two specific items. For further information about safety aspects in layout problems, the reader is referred to, for instance, Patsiatzis, Knight, and Papageorgiou (2004). Furthermore, in scheduling problems the $\boldsymbol{\delta}_{ij}$ and $\boldsymbol{\delta}_{ji}$ parameters may represent setup times where $\boldsymbol{\delta}_{ij}$ and $\boldsymbol{\delta}_{ji}$ are commonly unequal.

3.3.3. Orientation constraints

The size of each item, i, is given, and represented by $\mathbf{X}_i$ where $i = 1, \ldots, J$, and J is the total number of items. The optimal allocation of each item may, nonetheless, require items to rotate. By using the orientation constraints shown in Eqs. (5) and (6) the items are allowed to rotate in a certain number of dimensions. If the items are allowed to rotate in N-dimensions we obtain Eqs. (5) and (6) where $\mathbf{B}_i$ is a $N \times N$ matrix containing binary variables determining the orientation of each item:

$$\mathbf{X}_i = \mathbf{B}_i \mathbf{v}_i, \quad i = 1, 2, \ldots, J \tag{5}$$

$$\mathbf{B}_i \mathbf{e} = \mathbf{e}, \qquad \mathbf{B}_i^{\mathrm{T}} \mathbf{e} = \mathbf{e}, \quad i = 1, 2, \ldots, J \tag{6}$$

In Eq. (5) the given size of each item i is defined using the vectors $\mathbf{v}_i^{\mathrm{T}}$ according to: $\mathbf{v}_i^{\mathrm{T}} = (l_i, w_i, h_i, ...)$. Observe that by Eq. (6) the rotation is specified and that the sum of the binary variables on each row and each column of $\mathbf{B}_i$ is defined to be equal to one by Eq. (6). The binary variables on each row and each column can, thus, be considered as special ordered sets in a MILP solver. Observe further that if an item is not allowed to rotate in a certain coordinate direction, then the diagonal element in $\mathbf{B}_i$, corresponding to the coordinate, is defined to be equal to one.

3.3.4. Container constraints

The container constraints are defining the variables D_k, that is, whether a specific container is used or not, as seen in Eq. (7a). K is the total number of containers, and J is the total number of items:

$$\sum_{i=1}^{J} \beta_{ki} \leq J \cdot D_k, \quad k = 1, 2, \ldots, K \tag{7a}$$

Optionally, if a specified number of containers, N_K, is to be used, then Eq. (7b) should be used in addition to Eq. (7a):

$$\sum_{k=1}^{K} D_k = N_K \tag{7b}$$

In some cases it could be of interest to define Eq. (7b) as an inequality instead of an equality.

3.4. Additional constraints

Most allocation problems are large-scale combinatorial optimisation problems requiring significant computational effort for their solution. In order to reduce the combinatorial complexity, a number of optional formulation enhancement strategies are included in the proposed model. The additional constraints (8a)–(8c) and (9) should be included in the problem formulation

for objective function (2) only. The other additional constraints may be included, for both objective functions to reduce problem complexity, thus, cutting down the required CPU time. No case-specific degeneracy elimination cuts are presented in this paper, since the proposed model is a general purpose model for allocation problems in any given number of dimensions. For more information on degeneracy elimination procedures in allocation problems, the reader is referred to Sherali, Fraticelli, and Meller (2003), Castillo and Westerlund (2005) or Westerlund and Papageorgiou (2004).

3.4.1. Total and rectilinear distance constraints

If objective function (2) is to be used, the total distance of the items occupied in each container direction, and for each container should be defined. Since the distances are minimised, the constraints defining the distances, $\mathbf{s}_k$, may be formulated as shown in the following equation:

$$\mathbf{s}_k \geq \mathbf{x}_i + \tfrac{1}{2}\mathbf{X}_i + M \cdot (\beta_{ki} - 1) \cdot \mathbf{e},$$
$$i = 1, \ldots, J, \quad k = 1, \ldots, K \tag{8a}$$

If $K = 1$ or $N_K = 1$, then an additional condition can be given $\mathbf{s}_k$ according to the following equation:

$$\mathbf{s}_k \geq \gamma_k \cdot \mathbf{V}_k \cdot D_k, \quad k = 1, 2, \ldots, K \tag{8b}$$

where the parameter, γ_k, is given by the following equation:

$$\gamma_k = \frac{\sum_{i=1}^{J}(l_i \times w_i \times h_i \times \cdots)}{L_k \times W_k \times H_k \times \cdots} \tag{8c}$$

In cases where items in the same container are interconnected, the rectilinear distance, $\mathbf{d}_{ij}$, between the centroids is used as a basis for a material handling, pumping, or connection cost. The rectilinear distance between the centroid of two items, i, and j, is used in objective function (2). $\mathbf{d}_{ij} = 0$ if the items are not allocated in the same container. Constraints defining the rectilinear distances are given in the following equation:

$$\mathbf{d}_{ij} \geq \mathbf{x}_i - \mathbf{x}_j + M \cdot (G_{ij} - 1) \cdot \mathbf{e},$$
$$\mathbf{d}_{ij} \geq \mathbf{x}_j - \mathbf{x}_i + M \cdot (G_{ij} - 1) \cdot \mathbf{e},$$
$$\mathbf{e}^{\mathrm{T}} \cdot \mathbf{d}_{ij} \geq \tfrac{1}{2}[\min(l_i, w_i, \ldots) + \min(l_j, w_j, \ldots)]$$
$$+ M \cdot (G_{ij} - 1) \cdot \mathbf{e}, \quad 1 \leq i < j \leq J \tag{9}$$

Observe that $G_{ij} = 1$ if $K = 1$, or if Eq. (7b) is used with $N_K = 1$.

3.4.2. Symmetry-breaking constraints

Symmetry-breaking constraints are used to exclude equivalent symmetrical solutions, thereby reducing the CPU time needed for solution. For more information about different symmetry-breaking approaches the reader is referred to Westerlund and Papageorgiou (2004). The symmetry-breaking constraints used in the proposed model are designed to suit problems in N-dimensions without excluding the optimal solution. The considered symmetry-breaking constraints are bounding the first inserted item in each container to be placed in a determinate corner of the container, thereby breaking the symmetry:

$$\mathbf{x}_j \leq \tfrac{1}{2}\mathbf{U}_j + M \cdot \left(\sum_{i=1}^{j-1} G_{ij}\right) \cdot \mathbf{e}, \quad j = 1, 2, \ldots, J \tag{10a}$$

In Eq. (10a), M is an appropriate upper bound. Observe that the sum on the RHS is excluded when $j = 1$. In the specific case when $K = 1$ or if Eq. (7b) is used with $N_K = 1$ the symmetry-breaking constraints in Eq. (10a) could be replaced by the following equation:

$$\mathbf{x}_r \leq \mathbf{x}_s \tag{10b}$$

The indices, r and s, corresponds to different items. These items may be selected in different ways. They can be selected, for example, as the largest items or just (as default values) the first two items, that is, $r = 1$ and $s = 2$. The symmetry-breaking constraint (10b) has found to work very well in several two-dimensional examples as shown in Westerlund and Papageorgiou (2004).

3.4.3. Additional space constraints

Additional space constraints may be used to prevent attempts to allocate excessive number of items to the containers. The CPU time may significantly be decreased by using these constraints. The constraints shown in Eq. (11a) pre-determine the maximum space allowed to be occupied by items, inside each container, equal to or less than the total container space. The additional space constraints are most beneficial in multi container problems:

$$v_k \leq (L_k \times W_k \times H_k \times \cdots) \cdot D_k,$$
$$v_k = \sum_{i=1}^{J}(l_i \times w_i \times h_i \times \cdots) \cdot \beta_{ki}, \quad k = 1, 2, \ldots, K \tag{11a}$$

v_k is the summarised volume of the items in container k. In Eq. (11a), the parameters L_k, W_k, H_k, and so forth specify the given size of the considered container, and l_i, w_i, h_i, and so forth specify the given size of item i.

If identical containers, both in size and cost, exist, it might be worthwhile defining which of them is to be prioritised and filled first to maximum capacity. This procedure may substantially reduce the computational effort needed for solution. By using Eq. (11b), a container with a lower index, of the identical ones, is prioritised and filled to its maximum capacity prior to subsequent ones. q is the container with the lowest index, and N_q is the number of identical containers:

$$D_{k+1} \leq D_k, \qquad v_{k+1} \leq v_k, \quad k = q, \ldots, q + N_q - 2 \tag{11b}$$

Now two objective functions, and the essential constraints have been modelled and we are ready to illustrate the general formulation by some numerical examples.

4. Numerical examples

To illustrate the applicability of the proposed model, illustrative examples in a different number of dimensions are presented below.

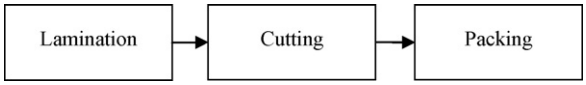

Fig. 5. Process flow-sheet for a flow-shop problem.

4.1. One-dimensional problems

A flow-shop problem with job dependent setup times is suitable for illustrating the proposed model applied on one-dimensional problems. A problem from Roslöf, Harjunkoski, Westerlund, and Isaksson (2002) will serve as an introductory illustrative problem for the proposed model.

4.1.1. A flow-shop problem

A flow-shop problem is characterised as a problem with a number of subsequent machines. Each job has to be processed on each machine. Furthermore, all jobs have to follow the same route (that is, they have to be processed first on machine 1, then machine 2, and so forth). After completion on one machine, a job joins the queue for the next machine. Usually, all queues are assumed to operate under the *First In First Out* (*FIFO*) discipline, that is, a job cannot "pass" another while waiting in queue, Pinedo (2002). The jobs in the queue may have different setup times and in such cases the job sequence needs to be optimised. As an illustrative example, a smaller version of an industrial flow-shop problem with job dependent setup times, presented in Roslöf et al. (2002), is solved and demonstrated using the proposed model. The problem data is taken from an industrial problem from the paper converting industry, and consists of a number of jobs which are to be "processed" in three steps: lamination, cutting, and packing, as seen in Fig. 5.

The jobs consist of different sized jumbo reels (large size paper reels) to be laminated with polythene and thereafter cut into product reels of specific sizes. The setup times are job dependent, since they are dependent on product quality and the size of the jumbo reels, as it requires longer setup times to change from a job with a narrow reel width to a job of broader width, and if the product quality changes.

A problem with 14 jobs is solved with the *N*-dimensional allocation formulation. The total number of variables for the problem is 429 including 197 binary variables. The total number of constraints used in the formulation is 729. The optimal solution of the problem is presented in Fig. 6 and Table 1. A big-*M* value of 10,000 was used.

If several jobs are processed for the same customer, a rectilinear cost parameter, $c_{ij} = 1$, has been used to penalise the time between these jobs. The optimal solution for the problem was reached in 0.17 CPUs when examining 21 nodes. The optimal value of the objective function was 4,482 and the make span of the schedule was 3,302. According to Roslöf et al. (2002) the solution time increases drastically with the number of jobs, when in addition to the make span, costs for tardiness, earliness, compactness, and so forth are included in the objective function. This was also observed with the given formulation. Processing times, setup times, and customers are given in Appendix B.

Table 1
Results for the 14-job flow-shop problem

Job	Processing time	Start time	Completion time	Customer
6	53	0	53	D
11	46	66	112	H
10	106	112	218	H
14	40	218	258	H
9	146	301	447	G
7	138	447	585	E
5	433	597	1,030	C
12	200	1,074	1,274	I
8	219	1,319	1,538	F
1	220	1,538	1,758	A
4	84	1,764	1,848	A
3	166	1,848	2,014	A
2	827	2,019	2,846	B
13	430	2,872	3,302	J

4.2. Two-dimensional problems

The proposed model is, further, applicable on a wide variety of two-dimensional problems. In Section 4.2.1 the solution of two strip-packing/cutting problems are presented as well as a single-floor Process Plant Layout problem in Section 4.2.2. Furthermore, there are numerous other kind of two-dimensional problems which could be solved with the proposed model, such as facility layout problems.

4.2.1. Strip-packing problems

The strip cutting/packing problem consists of cutting a large strip with a fixed width and unlimited length into smaller sub-rectangles, without violating the demand values imposed on each sub-rectangle, Hifi (1998). In Sawaya and Grossmann (2005), a nice cutting plane method for solving linear generalised disjunctive programming problems was proposed. The method was applied to the solution of some strip-packing problems where the solution efficiency of the method was compared to a Convex Hull and a Big-*M* formulation. In the strip-packing problem a given set of small rectangles are to be packed into a strip of fixed width but unknown length. The aim is to minimise the length of the strip while fitting all the rectangles without any overlap or rotation of rectangles. In Table 2, the results for a 12-rectangle strip-packing problem, obtained in Sawaya and Grossmann (2005) are shown. In this problem the ordered lengths and widths for the rectangles were given as follows: (1, 10), (2, 9), (3, 8), (4, 4), (5, 5), (9, 6), (7, 7), (6, 3), (5, 2), (12, 1), (3, 1), (2, 3). The given strip width is equal to 10.

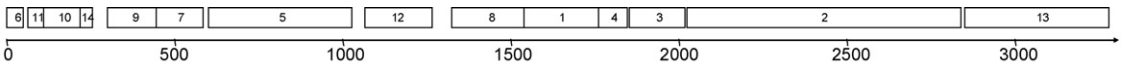

Fig. 6. Optimal production plan for 14-job flow-shop problem.

Table 2
Results for the 12-rectangle strip-packing problem (Sawaya & Grossmann, 2005)

	Number of constraints	Number of variables	Number of discrete variables	Relaxation	Optimal solution	Total number of nodes	CPUs
Convex hull	1,663	1,346	264	12	27	682,464	1286.4
Big-M	343	290	264	12	–	54,244,296	>10,800
Big-M + 87 cuts	430	290	264	12	27	72,677	27.5

Table 3
Results for the 12-rectangle strip-packing problem, with the proposed formulation

N-Dimensional allocation	Number of constraints	Number of variables	Number of discrete variables	Relaxation	Optimal solution	Total number of nodes	CPUs
+Eq. (8a)	559	381	265	12	27	27,881	18.5
+Eqs. (8a), (8b), (10a) and (11a)	587	382	265	24.5	27	16,352	8.8

4.2.1.1. Results with the proposed formulation. The strip-packing problem falls into the category of the N-dimensional allocation problems. The number of dimensions is, in this case, reduced to two, that is, $N=2$, and since the rectangles are to be cut from only one strip, the number of containers (in this case the strip) is equal to one, that is, $K=1$. The container constraints as given in Section 3.3.4 are, thus, not needed. In addition, since no rotation of the rectangles is allowed, the orientation constraints, as described in Section 3.3.3, are not needed. The variables and parameters in this two-dimensional allocation problem can be defined as follows. $\mathbf{x}^{\mathrm{T}}=(x, y)$, $\mathbf{X}^{\mathrm{T}}=(X, Y)$ and $\mathbf{v}^{\mathrm{T}}=(l, w)$. J is equal to the number of given rectangles, that is, $J=12$. The numerical values of the lengths and widths (l_i, w_i) of each rectangle, $i=1, \ldots, J$ were given above. The container (strip) dimensions are given by $L_1=30$, and $W_1=10$. The objective function can, in this case, be given by Eq. (2) where $C_1=0$, and $\bar{\mathbf{C}}_1^{\mathrm{T}}=(1, 0)$. No connections between any items exist, hence, no connection costs in the objective function. Furthermore, $\bar{\mathbf{c}}_i=0$ in Eq. (2). $M=100$ has been used in Eqs. (4a)–(4e), (8a) and (10a).

The results obtained with the N-dimensional allocation problem formulation are shown in Table 3, and the optimal strip-packing layout is shown in Fig. 7. In order to also illustrate the effect of the additional constraints, the results in the first row correspond to the case when only the main constraints and the total distance constraints in Eq. (8a) have been used. Row two illustrates the case when the additional constraints, Eqs. (8b), (10a) and (11a), have been used.

From Table 3 one can observe that a verified optimal solution to the strip-packing problem was obtained in 8.8 s by the given formulation and that 1/4 of the number of nodes were needed to be examined for the best solution when comparing it to the best

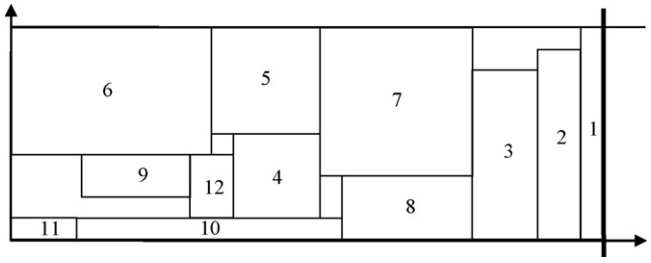

Fig. 7. Optimal solution for the 12-rectangle strip-packing problem.

formulation given by Sawaya and Grossmann (2005). Furthermore, it can be observed that the additional constraints give a very tight relaxation and also improve the solution efficiency of the problem. The results in Table 3 were obtained with CPLEX 8.0 (and default parameters) running on a 3.02 GHz Pentium PC.

Another strip-packing problem, also given in Sawaya and Grossmann (2005), containing 21 items was, additionally, solved. The item dimensions in the 21 item problem was the following: (1, 5), (2, 2), (3, 2), (2, 7), (5, 1), (6, 6), (5, 10), (4, 3), (3, 2), (9, 5), (4, 2), (1, 1), (2, 3), (3, 1), (2, 6), (2, 2), (1, 2), (2, 1), (2, 1), (1, 1), (1, 1) where the given strip width is 10 and the initial length is 30.

Solution results from the 21-rectangle problem reported by Sawaya and Grossmann (2005) are presented in Table 4, and the solution results using the N-dimensional allocation formulation are presented in Table 5.

In Fig. 8 the optimal strip-packing design for the 21-rectangle problem is presented. We found that for the 21-rectangle problem, as well as for the 12-rectangle problem, the N-dimensional allocation formulation yielded good solution efficiency.

Table 4
Results for the 21-rectangle strip-packing problem (Sawaya & Grossmann, 2005)

	Number of constraints	Number of variables	Number of discrete variables	Relaxation	Optimal solution	Total number of nodes	CPUs
Convex hull	5,272	4,244	840	9.1786	–	968,652	>10,800
Big-M	1,072	884	840	9	24	1,416,137	4093.4
Big-M + 62 cuts	1,134	884	840	9.1786	24	32,185	91.4

Table 5
Results for the 21-rectangle strip-packing problem with the proposed formulation

N-Dimensional allocation	Number of constraints	Number of variables	Number of discrete variables	Relaxation	Optimal solution	Total number of nodes	CPUs
+Eq. (8a)	1,639	1,137	840	9	24	33,780	97.39
+Eqs. (8a), (8b), (10a) and (11a)	1,685	1,138	840	22.5	24	13,512	36.3

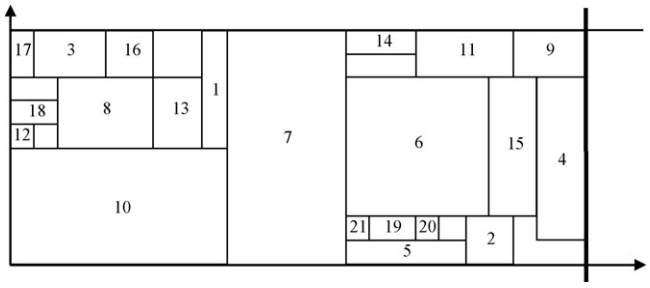

Fig. 8. Optimal solution for the 21-rectangle strip-packing problem.

4.2.2. *Single-floor Process Plant Layout problem*

Plant layout is concerned with the spatial arrangement of process plant and its interconnections, such as piping. A formal technique is any logical method that provides definitive information on relationships between items or numerical data on spacing distances, Mecklenburgh (1985). The Process Plant Layout (PPL) problem is one suitable application for the proposed model.

An illustrative PPL example considering an 11-equipment item batch plant from Papageorgiou and Rotstein (1998) is presented and solved with the *N*-dimensional allocation formulation. An initial flow-sheet for the considered problem, presented in Georgiadis, Schilling, Rotstein, and Macchietto (1997) is shown in Fig. 9. The problem data as well as the optimal allocation and orientation of each equipment item are presented in Tables 6 and 7. In Papageorgiou and Rotstein (1998), a layout solution, minimising the connectivity costs was given. The objective function value, obtained within a 5 percent margin of optimality, was 470.

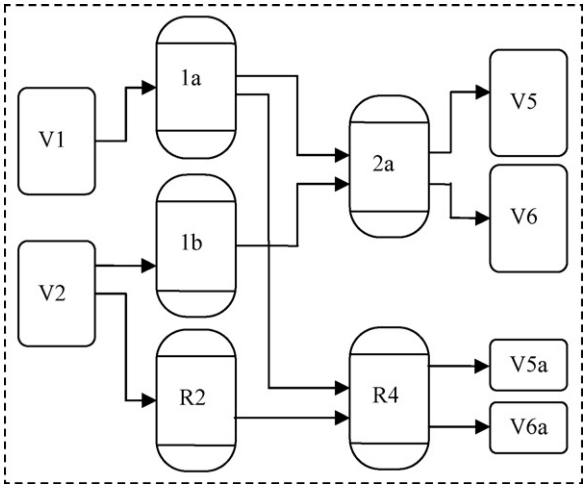

Fig. 9. Flow-sheet for PPL problem.

Table 6
Connection costs for PPL problem

Connection	Cost (rmu/m)
(V1, 1a)	1
(V2, 1b)	20
(V2, R2)	5
(1a, 2a)	10
(1a, R4)	1
(1b, 2a)	20
(2a, V5)	10
(2a, V6)	10
(R2, R4)	5
(R4, V5a)	1
(R4, V6a)	1

Using the proposed model, this particular Process Plant Layout problem is solved as a single-floor problem. For a multi-floor approach to the same problem, the reader is referred to Patsiatzis and Papageorgiou (2002).

The objective of the considered PPL problem is to minimise the connection costs between the equipment items. The optimal layout for the 11-equipment item Process Plant Layout problem is shown in Fig. 10. The total number of variables for the 11-equipment item problem is 497 including 265 binary variables, and the total number of constraints is 811. The additional constraints, Eqs. (9), (10b) and (11a), have been used in addition to the main constraints in the model. Big-$M = 40$ was used. Although the considered example only involves 11 items to be allocated, the problem is still rather difficult to solve. Feasible solutions can be obtained with modest computational requirements. Quite much CPU time was, however, needed to obtain a verified optimal solution, for this particular example, even though a tight relaxation is given with the model. In Table 8,

Table 7
Footprint sizes and optimal item allocation for PPL problem

Equipment	Item size		Optimal location		Optimal orientation	
	l_i	w_i	x_i	y_i	X_i	Y_i
V1	5.0	3.0	5.0	1.5	5.0	3.0
V2	6.0	6.0	11.0	17.0	6.0	6.0
1a	6.0	6.0	5.0	6.0	6.0	6.0
1b	5.0	5.0	11.0	11.5	5.0	5.0
2a	6.0	6.0	11.0	6.0	6.0	6.0
R2	4.5	4.5	5.5	17.0	4.5	4.5
R4	5.0	5.0	5.5	12.25	5.0	5.0
V5	5.0	3.0	11.0	1.5	5.0	3.0
V6	6.0	6.0	17.0	6.0	6.0	6.0
V5a	2.0	1.0	2.5	12.25	1.0	2.0
V6a	3.0	2.0	1.0	12.25	2.0	3.0

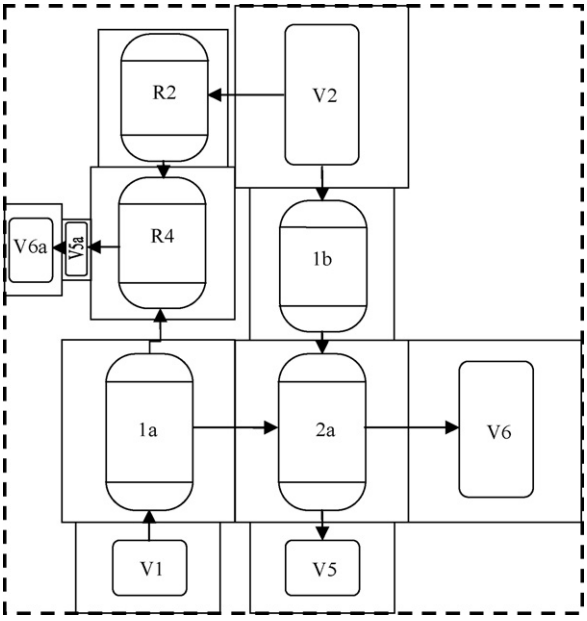

Fig. 10. Optimal layout for 11-unit PPL problem.

solution results when solving the problem with different margins of optimality are given.

The flow-sheet for the optimal solution, with minimum connection costs of 455, is illustrated in Fig. 10. The optimal item allocation is given in Table 7.

4.3. Three-dimensional problems

In the literature, container loading problems, palletisation problems, and related problem types have received a lot of attention. Hence, an illustrative problem where the proposed model is applied on a three-dimensional packing problem is presented in Section 4.3.1.

4.3.1. Three-dimensional packing problem

A mixed-sized box packing problem presented in Westerlund et al. (2005a) is solved using the proposed model. The problem consists of 13 boxes and 4 containers. Each of the boxes should be allocated inside one of the containers. The objective is to minimise the summarised container costs, according to Eq. (1). The problem data as well as the optimal box allocation and orientation for this example problem are given in Tables 9 and 10.

The optimal solution, 190, was reached in 17.2 CPUs after 8,131 nodes using objective function (1). In this example the total number of variables was 840 including 719 binary variables, and the number of constraints was 1,355 in the N-dimensional allocation formulation including the additional constraints. Big-$M = 100$ was used in the optimisation. The optimal solution using objective function (1) does not, despite the fact, minimise the item allocation in any coordinate direction, as long as only the total container cost is minimised. This means that a solution where some of the items may be "floating" above the container floor can be obtained, since the objective function is not forcing boxes to take minimum possible z-coordinate. The solution is, consequently, not taking any gravity force, forcing boxes down in z-direction, into consideration.

Table 8
Solution data for the 11-item Process Plant Layout problem

MIP gap (percent)	Relaxation	Obj. function	Nodes	CPUs
5	451.5	472.5	4,147	4.1
3	451.5	459.5	52,241	47.2
1	451.5	455.5	67,870	59.8
0	451.5	455.0	455,362	332.4

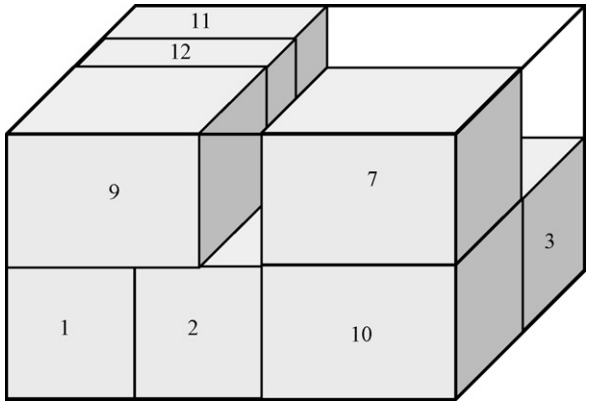

Fig. 11. Packing pattern for container 3.

Using objective function (2), and simultaneously minimising the item allocation in the z-coordinate prevents "floating" boxes but requires more CPU time. This procedure could in practice mean that gravity is taken into consideration. As an alternative, solution data from the first problem, using objective function (1), may be used as input data for the reallocation of items in the height, z, dimension of each container. Using this alternative way of solving the problem, the problem is first solved using objective function (1) and the solution of this problem is then used in the following sub-problems, where the items are reallocated using objective function (2). Each container is in the second step considered as a separate problem. Using this procedure, the first sub-problem was solved in 0.1 CPUs and the second sub problem in 0.2 CPUs. The solution data is shown in Table 9. In Figs. 11 and 12, the optimal packing pattern for the 13-box problem is shown, where objective function (2) has been used to prevent boxes from "floating" in the z-coordinate.

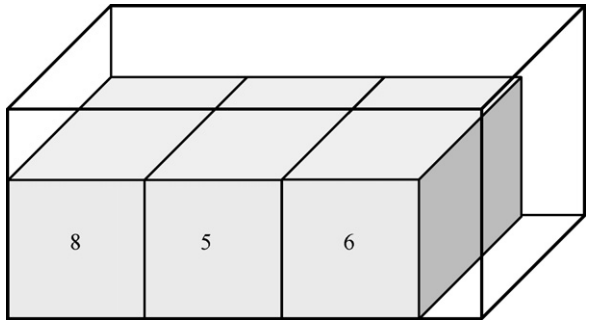

Fig. 12. Packing pattern for container 1.

Table 9
Box dimensions and optimal packing pattern for packing problem

Box	Size			Optimal orientation			Optimal allocation			
	L_i	W_i	H_i	X_i	Y_i	Z_i	x_i	y_i	z_i	Container
1	2	2	2	2	2	2	1.0	1.0	1.0	3
2	2	2	2	2	2	2	1.0	3.0	1.0	3
3	2	2	2	2	2	2	3.0	6.0	1.0	3
4	2	2	2	2	2	2	3.0	4.0	2.0	3
5	2	2	3	3	2	2	1.5	3.0	1.0	1
6	2	2	3	3	2	2	1.5	5.0	1.0	1
7	2	2	3	2	3	2	1.0	5.5	3.0	3
8	2	2	3	3	2	2	1.5	1.0	1.0	1
9	2	2	3	2	3	2	1.0	1.5	3.0	3
10	2	2	3	2	3	2	1.0	5.5	1.0	3
11	3	3	1	1	3	3	3.5	1.5	2.5	3
12	3	3	1	1	3	3	2.5	1.5	2.5	3
13	1	2	5	2	5	1	3.0	2.5	0.5	3

Observed that items 4, and 13 are not visible in container 3 with the used projection.

4.4. Four-dimensional problems

Two four-dimensional problems, where a number of items are to be allocated in three spatial dimensions and in one time dimension, are finally illustrated in Section 4.4.1. Many different kinds of optimisation problems may be formulated and solved as four-dimensional allocation problems. The easiest way of illustrating a four-dimensional problem is to consider the first three dimensions as spatial dimensions and the fourth dimension as a time dimension.

4.4.1. Four-dimensional processing problem

The four-dimensional problems presented below consist of a number of items which are to be allocated inside a given number of containers with limited spatial and time dimensions. Each item should be allocated inside some of the containers subject to the constraints presented above. In the four-dimensional case, items might overlap each other in all space dimensions as long as the time dimension is left un-violated and vice versa. The items are allowed to rotate in the three space dimensions but not in the time direction. Thus, the fourth diagonal element of the matrixes $\mathbf{B}_i$ in Eqs. (5) and (6) is defined to be equal to 1. The first problem consists of 16 items and 1 container. Each item should be allocated inside a container for a certain period of time. This time may be considered as specific processing times, given in Table B3 in Appendix B. The objective is to minimise the total time needed for processing of all items. The second problem consists of 16 items and 2 containers. The number of variables in the first problem is 1,547 including 1,233 binary variables, and the number of constraints is 2,044. The corresponding numbers for the second problem are 1,566, 1,370, and 2,302, respectively. Big-$M = 5$ was used in both problems.

Table 10
Container dimensions and costs for packing problem

Container	Length	Width	Height	Cost
1	3	7	3	80
2	3	7	3	80
3	4	7	4	110
4	4	7	4	110

The illustrative examples might represent problems where a number of items, for example, are to be painted, sterilised, heated up, and so forth for a certain period of time inside a process unit (container) with limited dimensions and availability in time. The optimal time allocation for the 16 items, 1 container problem is shown in Fig. 13 and the optimal time allocation for the 16 items, 2 container problem is shown in Fig. 14. The optimal spatial allocation and orientation of the items in the containers are given in Tables B4–B6 in Appendix B. The first problem was solved in 450.6 CPUs after 120,029 nodes with an objective function value of 2.0. The second problem was solved in 30.3 CPUs after 4,861 nodes with an objective function value of 3.0. In both problems the container cost, C_k, is equal to 1 for all available containers. The elements of the cost vector, $\bar{\mathbf{C}}_k$, are 0, 0, 0, and 0.1 for all containers, thus, only costs for the availability in time are used.

Note that an equivalent solution value could have been reached for the second four-dimensional problem by allocating items 2, 4, 5, 6, and 9 inside container 2 after $t = 2$, thereby

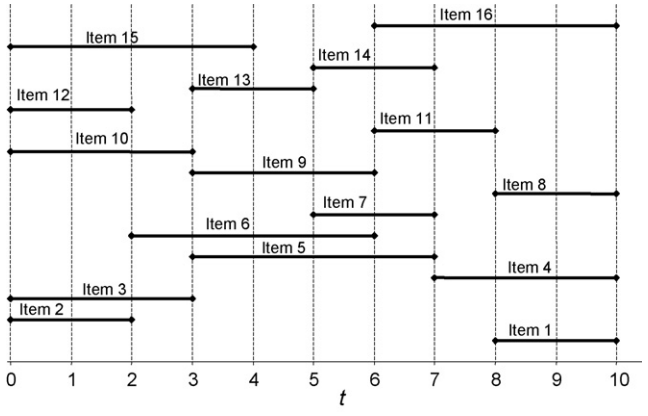

Fig. 13. Time allocation of boxes for the four-dimensional allocation problem with 16 items and 1 container.

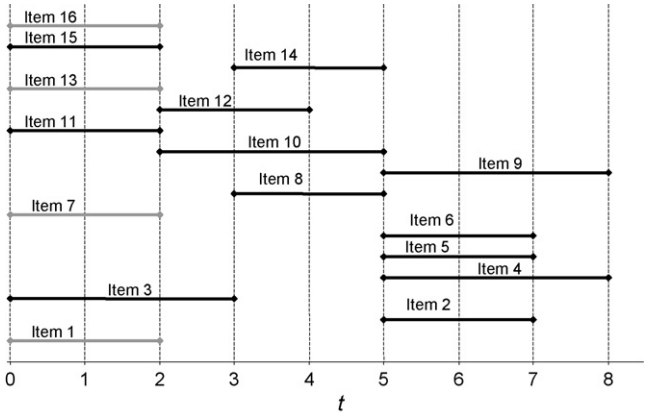

Fig. 14. Time allocation of boxes for the four-dimensional allocation problem with 16 items and 2 containers (grey lines representing items allocated in container 2 and black lines items allocated in container 1).

reducing the total process span. However, since the objective in the given problem was to minimise the summarised processing time for both containers, the solution presented above also gives the same optimal objective function value.

5. Discussion

In this paper a general purpose MILP model for allocation problems in any given number of dimensions is presented. The model is called the *N*-dimensional allocation formulation. The considered formulation is concerned with the allocation of items in an *N*-dimensional space. To demonstrate the model, the proposed formulation is described and applied to a number of illustrative examples in a different number of dimensions. One-dimensional flow-shop scheduling problems, two-dimensional strip-packing/cutting problems, two-dimensional Process Plant Layout problems, three-dimensional packing problems, and four-dimensional problems are presented and solved using the *N*-dimensional allocation formulation. For a specified number of dimensions more efficient, special purpose, models may be created. The given *N*-dimensional allocation model also offers, however, in such cases, a general platform as a basis for the model. The novelty of the formulation is its applicability to a wide range of problems in any given number of dimensions. A number of previously published problems are, furthermore, solved efficiently using the proposed model.

The prevalent challenge for the proposed model is the computational complexity which tend to rise dramatically with increasing number of items and dimensions. In cases where the problem-size rise beyond control, the proposed *N*-dimensional allocation model may be used in combination with a decomposition-based approach together with problem-specific logical constraints to reach feasible solutions within a reasonable time.

Acknowledgement

The first author gratefully acknowledges the financial support from the Academy of Finland.

Appendix A

A.1. Overlapping of N-dimensional items

First consider an item i defined by a set $C_i = \{\mathbf{x}|\mathbf{c}_i^{\mathrm{L}} \leq \mathbf{x} \leq \mathbf{c}_i^{\mathrm{U}}\}$. Then, for two items, i and j, an overlapped region is defined by an non-empty set $O_{i,j} = C_i \cap C_j$.

Proposition. *Two N-dimensional items, i and j, are allowed to overlap in up to* $N-1$ *dimensions without having an overlapped region.*

Proof. Consider any point $\mathbf{x}^0 \in C_i$. Allowing overlapping of the item i with an item j in at most $N-1$ dimensions, there must exist at least one index, s (corresponding to an element of $\mathbf{x}^0$) such that $x_s \notin [c_{j,s}^{\mathrm{L}}, c_{j,s}^{\mathrm{U}}]$. Thus, $\mathbf{x}^0 \in C_j$, and since $O_{i,j} = C_i \cap C_j$, it implies that $\mathbf{x}^0 \notin O_{i,j}$. We can therefore conclude that two *N*-dimensional items, i and j, are allowed to overlap in up to $N-1$ dimensions without having an overlapped region. □

Consider the constraints in Eqs. (4a)–(4e). When two items are allocated in the same container, then $G_{i,j} = 1$. Thus, we obtain from (4d), $\mathbf{e}^{\mathrm{T}}\mathbf{P}_{i,j} \leq N-1$, which allows not more than $N-1$ of the binary variables in the vector $\mathbf{P}_{i,j}$ to be equal to one. Furthermore, when an element in the vector $\mathbf{P}_{i,j}$ is equal to one, we obtain for this particular dimension $P_{i,j} + Q_{i,j} \geq 1$ and $P_{i,j} - Q_{i,j} + 1 \geq 1$ for any value of the binary variable $Q_{i,j}$. Thus the overlapping constraints are always relaxed (with the big-M on the RHS) in up to $N-1$ dimensions with up to $N-1$ elements, equal to one, in the vector $\mathbf{P}_{i,j}$. On the other hand, when an element $P_{i,j} = 0$, then $P_{i,j} + Q_{i,j} = Q_{i,j}$ and $P_{i,j} - Q_{i,j} + 1 = 1 - Q_{i,j}$. Thus, depending on the value of the binary variable $Q_{i,j}$ one of the overlapping constraints, in this particular dimension, will never be relaxed (with the big-M in the RHS). Thus, the overlapping constraints in (4a)–(4e) fulfil the properties in the proposition.

Appendix B

See Tables B1–B6.

Table B1
Processing times and customer specification for the flow-shop problem

Job	Customer	Processing time
1	A	220
2	B	827
3	A	166
4	A	84
5	C	433
6	D	53
7	E	138
8	F	219
9	G	146
10	H	106
11	H	46
12	I	200
13	J	430
14	H	40

Table B2
Setup times for the flow-shop problem

j	i													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	–	0	7	0	0	5	9	0	0	0	0	0	0	0
2	15	–	5	0	0	2	7	0	0	0	0	0	0	15
3	0	0	–	0	0	15	5	0	0	0	0	0	0	0
4	6	7	8	–	15	7	10	0	0	0	0	0	0	6
5	8	8	10	5	–	9	12	0	0	15	0	0	0	8
6	0	0	5	0	0	–	7	0	0	0	0	0	0	0
7	0	0	15	0	0	0	–	0	0	0	0	0	0	0
8	55	45	58	54	52	56	60	–	45	52	52	45	50	55
9	43	44	45	42	42	44	47	0	–	40	40	0	25	43
10	9	9	11	7	5	10	13	0	0	–	0	0	0	9
11	11	12	14	10	5	13	15	0	0	0	–	0	0	11
12	45	46	47	44	44	46	50	0	5	42	42	–	30	45
13	26	26	28	25	24	27	45	0	0	22	22	0	–	26
14	0	0	7	0	0	5	9	0	0	0	0	0	0	–

Rows represent preceding jobs and columns succeeding jobs.

Table B3
Data for the four-dimensional allocation problem with 16 items and 1 container

Item	Length	Width	Height	Time
1	2	2	2	2
2	2	2	2	2
3	3	3	3	3
4	3	3	3	3
5	3	2	2	4
6	3	2	2	4
7	3	3	2	2
8	3	3	2	2
9	4	2	2	3
10	4	2	2	3
11	4	3	2	2
12	4	3	2	2
13	4	3	3	2
14	4	3	3	2
15	2	2	1	4
16	2	2	1	4
Container				1
Length				5
Width				5
Height				5
Time				26

Table B4
Solution data for the four-dimensional allocation problem with 16 items and 1 container

Item	x_i/X_i	y_i/Y_i	z_i/Z_i	t_i/T_i
1	1.0/2.0	1.0/2.0	1.0/2.0	9.0/2.0
2	1.0/2.0	1.0/2.0	1.0/2.0	1.0/2.0
3	3.5/3.0	1.5/3.0	3.5/3.0	1.5/3.0
4	3.5/3.0	3.5/3.0	3.5/3.0	8.5/3.0
5	3.5/3.0	4.0/2.0	1.0/2.0	5.0/4.0
6	1.0/2.0	4.0/2.0	1.5/3.0	4.0/ 4.0
7	3.5/3.0	1.5/3.0	1.0/2.0	6.0/2.0
8	1.5/3.0	1.0/2.0	3.5/3.0	9.0/2.0

Table B4 (*Continued*)

Item	x_i/X_i	y_i/Y_i	z_i/Z_i	t_i/T_i
9	1.0/2.0	2.0/4.0	4.0/2.0	4.5/3.0
10	4.0/2.0	2.0/4.0	1.0/2.0	1.5/3.0
11	1.0/2.0	2.0/4.0	3.5/3.0	7.0/2.0
12	1.0/2.0	3.5/3.0	2.0/4.0	1.0/2.0
13	3.5/3.0	1.5/3.0	2.0/4.0	4.0/2.0
14	3.5/3.0	2.0/4.0	3.5/3.0	6.0/2.0
15	4.0/2.0	4.5/1.0	4.0/2.0	2.0/4.0
16	1.0/2.0	4.5/1.0	4.0/2.0	8.0/4.0
Objective function value				2.0
CPU time				450.6
Nodes				120,029

Table B5
Data for the four-dimensional allocation problem with 16 items and 2 containers

Item	Length	Width	Height	Time
1	2	2	2	2
2	2	2	2	2
3	3	3	3	3
4	3	3	3	3
5	3	2	2	4
6	3	2	2	4
7	3	3	2	2
8	3	3	2	2
9	4	2	2	3
10	4	2	2	3
11	4	3	2	2
12	4	3	2	2
13	4	3	3	2
14	4	3	3	2
15	2	2	1	4
16	2	2	1	4

Container	Length	Width	Height	Time
1	5	5	5	8
1	5	5	5	8

Table B6
Solution data for the four-dimensional allocation problem with 16 items and 2 containers

Item	x_i/X_i	y_i/Y_i	z_i/Z_i	Container	t_i/T_i
1	1.0/2.0	2.5/2.0	1.0/2.0	2	1.0/2.0
2	4.0/2.0	4.0/2.0	1.0/2.0	1	6.0/2.0
3	1.5/3.0	1.5/3.0	3.5/3.0	1	1.5/3.0
4	3.5/3.0	1.5/3.0	1.5/3.0	1	6.5/3.0
5	1.0/2.0	1.5/3.0	1.0/2.0	1	6.0/4.0
6	1.5/3.0	4.0/2.0	1.0/2.0	1	6.0/4.0
7	3.5/3.0	1.0/2.0	3.5/3.0	2	1.0/2.0
8	1.5/3.0	1.0/2.0	3.5/3.0	1	4.0/2.0
9	2.0/4.0	1.0/1.0	4.0/2.0	1	6.5/3.0
10	4.0/2.0	1.0/2.0	2.0/4.0	1	3.5/3.0
11	3.0/4.0	3.5/3.0	1.0/2.0	1	1.0/2.0
12	1.5/3.0	3.0/4.0	1.0/2.0	1	3.0/2.0
13	3.5/3.0	3.5/3.0	2.0/4.0	2	1.0/2.0
14	2.0/4.0	3.5/3.0	3.5/3.0	1	4.0/2.0
15	4.5/1.0	4.0/2.0	4.0/2.0	1	1.0/2.0
16	4.0/2.0	4.0/2.0	4.5/1.0	2	1.0/2.0
Objective function value					3.0
CPU time					30.3
Nodes					4,861

References

Amorese, L., Cena, V., & Mustacchi, C. (1991). A heuristic for the compact location of process components. *Chemical Engineering Science*, *32*, 119–124.

Armour, G. C., & Buffa, E. S. (1963). A heuristic algorithm and simulation approach to relative allocation of facilities. *Management Science*, *9*, 294–309.

Barbosa-Póvoa, A. P., Mateus, R., & Novais, A. Q. (2002). Optimal 3D layout of industrial facilities. *International Journal of Production Research*, *40*(7), 1669–1698.

Castillo, I., Westerlund, J., Emet, S., & Westerlund, T. (2005). Optimization of block layout design problems with unequal areas: A comparison of MILP and MINLP optimization methods. *Computers and Chemical Engineering*, *30*(1), 54–69.

Castillo, I., & Westerlund, T. (2005). An ε-accurate model for optimal unequal-area block layout design. *Computers and Operations Research*, *32*, 429–447.

Georgiadis, M. C., Schilling, G., Rotstein, G. E., & Macchietto, S. (1997). Optimal layout design in multipurpose batch plants. *Industrial and Engineering Chemistry Research*, *36*, 4852.

Gunn, D. J., & Al-Asadi, H. D. (1987). Computer aided layout of chemical plant: A computational method and case study. *Computer Aided Design*, *19*, 131–140.

Hifi, M. (1998). Exact algorithms for the guillotine strip cutting/packing problem. *Computers and Operations Research*, *25*(11), 925–940.

Jayakumar, S., & Reklaitis, G. V. (1994). Chemical plant layout via graph partitioning. I. Single level. *Computers and Chemical Engineering*, *14*, 441–458.

Jernström, P., Westerlund, J., & Papageorgiou, L. G. (2004). An item-based decomposition approach to process plant layout problems. In *Proceedings of the Fourth IASTED conference on modelling simulation and optimization* (pp. 37–42).

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

Mecklenburgh, J. C. (1985). *Process plant layout*. UK: The Institution of Chemical Engineers.

Meller, R. D., & Gau, K. Y. (1996). The facility layout problem: Recent and emerging trends and perspectives. *Journal of Manufacturing Systems*, *15*, 351–366.

Papageorgiou, L. G., & Rotstein, G. E. (1998). Continuous domain mathematical models for optimal process layout. *Industrial and Engineering Chemistry Research*, *37*, 226–231.

Patsiatzis, D. I., Knight, G., & Papageorgiou, L. G. (2004). An MILP approach to safe process plant layout. *Chemical Engineering Research and Design*, *82*(A5), 579–586.

Patsiatzis, D. I., & Papageorgiou, L. G. (2002). Optimal multi-floor process plant layout. *Computers and Chemical Engineering*, *26*, 575–583.

Patsiatzis, D. I., & Papageorgiou, L. G. (2003). Efficient solution approaches for the multifloor process plant layout problem. *Industrial and Engineering Chemistry Research*, *42*, 811–824.

Penteado, F. D., & Ciric, A. R. (1996). An MINLP approach for safe process plant layout. *Industrial and Engineering Chemistry Research*, *37*, 1354–1361.

Pinedo, M. (2002). *Scheduling. Theory, algorithms and systems* (2nd ed.). USA: New York University.

Roslöf, J., Harjunkoski, I., Westerlund, T., & Isaksson, J. (2002). Solving a large-scale industrial scheduling problem using MILP combined with a heuristic procedure. *European Journal of Operational Research*, *138*(1), 29–42.

Sawaya, N. W., & Grossmann, I. E. (2005). A cutting plane method for solving linear generalized disjunctive programming problems. *Computers and Chemical Engineering*, *29*(9), 1891–1913.

Sherali, H. D., Fraticelli, B. M. P., & Meller, R. D. (2003). Enhanced model formulations for optimal facility layout. *Operations Research*, *51*(4), 629–644.

Suzuki, A., Fuchino, T., Muraki, M., & Hayakawa, T. (1991). Method of determining the floor for sitting of each equipment unit of a multipurpose batch plant. *Kagaku Kogaku Ronbunshu*, *17*, 1110–1117.

Tsai, R. D., Malmstrom, E. M., & Kuo, W. (1993). Three dimensional palletization of mixed box sizes. *IIE Transactions*, *25*, 64–75.

Westerlund, J., & Papageorgiou, L. G. (2004). Improved performance in process plant layout problems using symmetry-breaking constraints. In *Proceedings of the FOCAPD on discovery through product and process design* (pp. 485–488).

Westerlund, J., Papageorgiou, L. G., & Westerlund, T. (2005a). A problem formulation for optimal mixed-sized box packing. In *Proceedings of the 15th European symposium on computer aided process engineering, Vol. 1* (pp. 913–918). Elsevier.

Westerlund, J., Papageorgiou, L. G., & Westerlund, T. (2005b). A problem formulation for *N*-dimensional allocation. *Chemical Engineering Transactions*, *6*, 185–190.