CHEMICAL ENGINEERING TRANSACTIONS, Volume 11, 2007
Edited by Sauro Pierucci
 ISBN 978-88-95608-00-6

An Interactive Optimisation Tool for Allocation Problems

Fredrik Bonäs, Joakim Westerlund and Tapio Westerlund
Process Design Laboratory, Faculty of Technology,
Åbo Akademi University, Turku 20500, Finland

This paper presents an interactive tool (MINDAT) for solving allocation problems in up to four dimensions. Typically such problems are one-dimensional multi-product scheduling problems, two-dimensional cutting problems, three-dimensional packing problems and four-dimensional logistical problems. The allocation problems are formulated as mixed integer linear programming (MILP) models or mixed integer non-linear programming (MINLP) models depending on the nature of the problem. The software consists of a Graphical User Interface (GUI) into which problem data is feed and where results are displayed, and a link to the commercial optimisation solver ILOG CPLEX 10.0.1.

1. Introduction

The allocation problems are modelled with a fundamental problem formulation for *N*-dimensional allocation problems presented in (Westerlund, 2005), with some slight modifications. In these models items of *N*-dimensions are to be allocated in one or more arrays, called containers, of the same dimension. A container may represent a limited one-dimensional array, an unlimited space in time, a two-dimensional area or a three-dimensional volume. If a three-dimensional item also has a fourth time-dimension, determining its availability in time, it may be considered as a four-dimensional item. The model handles all items as linear, rectangular, cubic etc. If all directional sizes of the items are given, i.e. width, height and depth, a linear formulation is used. In this case the model consists of an objective function and four main types of constraints. These constraints determine the items position inside the containers, prevent overlapping between the items, allow the items to rotate in any given direction and define if a certain container is used or not. Additional constraints concerning connection costs, costs for placement of an item in any given direction and costs for used space in a certain direction in a container may also be used. For problems where the area or volume is given instead of the specific directional sizes of the items a corresponding non-linear formulation is used. This formulation results in a convex MINLP-model that can be solved to global optimality using a simplification of the Extended Cutting Plane Method described in (Westerlund and Pörn, 2002).

2. MINDAT

The program described in this paper is called MINDAT, which stands for Mixed Integer *N*-dimensional Allocation Tool. The purpose of the software is that users easily shall be able to solve allocation problems of different nature without deep insight in mixed integer mathematical programming.

3. Problem formulation

The general model for an *N*-dimensional problem is formulated by using vector notations in order to make the formulation as compact as possible. Thus an *N*-dimensional item *i* is represented by its coordinate-vector $\mathbf{x}_i$ (defining the centroid coordinates of the item), where $\mathbf{x}_i^T = (x_i, y_i, z_i, t_i ...)$ and $N = \dim(\mathbf{x}_i)$. The size of the item *i* is, in each direction defined by the vector $\mathbf{X}_i$, where $\mathbf{X}_i^T = (X_i, Y_i, Z_i, T_i ...)$, X_i, Y_i, Z_i and T_i represents the length, height, width and availability in time of the item. Note that the elements of $\mathbf{X}_i$ are variables. In case the size of an item is defined by given length, height, width etc. The elements of $\mathbf{X}_i$, in the different directions, are selected through so called orientation constraints from the size-vector $\mathbf{v}_i^T = (l_i, h_i, w_i ...)$ including given sizes for the item *i.* If the size of an item is defined by an area or volume the elements of $\mathbf{X}_i$ are defined through size constraints including upper and lower bounds of the elements in $\mathbf{X}_i$. The size of the container in which item *i* is allocated is defined by the vector $\mathbf{U}_i$. Each container *k* has a vector $\mathbf{V}_k^T = (L, H, W, ...)$ defining its size. The values of the variables in $\mathbf{U}_i$ are selected from one of the vectors $\mathbf{V}_k$.

3.1 An MILP formulation of an *N*-dimensional allocation problem

In this case the size, in different directions, of every item is predefined for each dimension *N*, but the item may still be allowed to rotate. The total number of items is *J*, and the total number of containers is *K*.

3.1.1 The Objective Function

The objective is to minimise a cost function. The cost can be dependent on which container is used, total space used in a container in a certain direction, rectilinear distance between items and the centroid coordinate of each item. The objective function used in MINDAT is given in equation (1).

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

In equation (1) C_k is the given cost for container *k*, and D_k is a binary variable defining if container *k* is used or not. The vector $\mathbf{C}_k$ contains costs connected to the total used space, $\mathbf{s}_k$ in each coordinate direction, and $\mathbf{c}_{ij}$ is a vector with costs for the rectilinear distance, $\mathbf{d}_{ij}$ between items *i* and *j*. Finally $\mathbf{c}_i$ defines a costs connected to the centroid coordinates $\mathbf{x}_i$ for each item *i*. Elements of this latter cost parameter vector can, for example, be used as “gravity parameters” to avoid “flying” boxes in packing problems. Note that the objective function can be facilitated simply by leaving out one or more, but not all, of the terms in equation (1).

3.1.2 Constraints

In this section we consider the constraints involved in the formulation. Some of the constraints are given explicitly while the reader is referred to (Westerlund, 2005) for a complete set of constraints. ***Space constraints*** are used to make sure that no items are allocated outside a container, and that all items are allocated in one container. ***Overlap Prevention Constraints*** are used to prevent items to overlap. Note that two items may overlap each other in *N*-1 dimensions, (Westerlund, 2005). The constraints shown in equation (2) are slightly modified from the overlap prevention constraints in (Westerlund, 2005).

$$\left.\begin{aligned}
&\frac{\mathbf{X}_i+\mathbf{X}_j}{2}+\boldsymbol{\delta}_{ij} \le \left(\mathbf{x}_i-\mathbf{x}_j\right)+M\left(\mathbf{e}-\mathbf{P}_{ij}\right)\\
&\frac{\mathbf{X}_i+\mathbf{X}_j}{2}+\boldsymbol{\delta}_{ji} \le \left(\mathbf{x}_j-\mathbf{x}_i\right)+M\left(\mathbf{e}-\mathbf{Q}_{ij}\right)\\
&\mathbf{e}^T\left(\mathbf{P}_{ij}+\mathbf{Q}_{ij}\right) \ge G_{ij}\\
&\beta_{ki}+\beta_{kj} \le G_{ij}+1
\end{aligned}\right\}
\begin{aligned}
&1 \le i < j \le J\\
&k = 1,2,...,K\\
&K \ne 1
\end{aligned}
\qquad (2)$$

The binary variable G_{ij} in equation (2) equals one if the items *i* and *j* are located in the same container; zero otherwise. The parameter M is an appropriate upper bound. $\mathbf{P}_{ij}$ and $\mathbf{Q}_{ij}$ are vectors of binary variables that are used to prevent overlap. Since according to (Westerlund, 2005) two items may overlap each other in *N*-1 dimensions, only one of the binaries in the vector $\mathbf{P}_{ij}$ and $\mathbf{Q}_{ij}$ need to be equal to one (if the two items are in the same container). When a binary variable in $\mathbf{P}_{ij}$ or $\mathbf{Q}_{ij}$ is zero the corresponding inequality is relaxed. $\mathbf{e}$ is a unit vector of length *N*, $\mathbf{e}^T = (1,1,1,...)$, and $\boldsymbol{\delta}_{ij}$ is a vector of parameters defining the minimum distance or setup-time between item *i* and *j*. β_{ki} is a binary variable equal to one if item *i* is located in container *k* and zero otherwise. ***Orientation Constraints*** **are** used to allow items to rotate. The ***container constraints*** are used to define if a certain container is used or not. Since we in the objective function minimise the distances ($\mathbf{s}_k$) occupied by items in each direction in container *k* do we also need additional constraints defining the distance-vector. Such constraints are given in (Westerlund, 2005). If only one container is available the distance can be expressed in a more condensed form, see (Westerlund, 2005). If we have a connection cost between items in the same container, the rectilinear distance between these items needs to be calculated. ***Rectilinear distance constraints*** for this purpose are also given in (Westerlund, 2005). If items *i* and *j* are in the same container the vector $\mathbf{d}_{ij}$ in the objective function (2) defines the rectilinear distance, otherwise $\mathbf{d}_{ij}$ is zero. ***Symmetry-breaking constraints*** are used to prevent equivalent symmetrical solutions and, thus, also to reduce the solution time for the problem. This can be done for example by forcing the first inserted item in each container into a certain corner of the container. The reader is referred to (Westerlund, 2005) for a more detailed discussion of the symmetry-breaking constraints. To prevent items to overload a

container (and also computationally to reduce solution time) ***capacity constraints*** has additionally been formulated as shown in (Westerlund, 2005). If several identical containers exist, in size and cost, one can furthermore add constraints determining which of these should be allocated first. By using these constraints a container with a lower index of the identical ones is filled to its maximum first.

3.2 Optimisation of Block/Box Layout Design Problem

In the block layout design problem, (Castillo et. al. 2005) or box layout design problem in 3 and 4 dimensional cases, the items are represented by areas or volumes instead of items with given side-lengths. In this case we will only consider one container which is represented by an area or a volume with given size. The objective function and constraints described in chapter 3.1 are used also in this case, although since the number of containers is always equal to one the constraints concerning containers are modified to fit this case. The orientation constraints can be removed since the vector $\mathbf{X}_i$ now consists of variables defined by upper and lower bounds, and the size of the item. These bounds are in the 2-dimensional case defined according to (Castillo et. al. 2005) as,

$$w_i^{up} = \min\left\{\sqrt{a_i\alpha_i}, W\right\} \quad h_i^{up} = \min\left\{\sqrt{a_i\alpha_i}, H\right\} \tag{3a}$$

$$w_i^{low} = \max\left\{\sqrt{\frac{a_i}{\alpha_i}}, \frac{a_i}{h_i^{up}}\right\} \quad h_i^{low} = \max\left\{\sqrt{\frac{a_i}{\alpha_i}}, \frac{a_i}{w_i^{up}}\right\} \tag{3b}$$

The parameter α_i in (3a)-(3b) is an aspect ratio defined by $\alpha_i = \max\{l_i, w_i\} / \min\{l_i, w_i\}$, W and H are the width and height of the container and a_i is the area of the item *i*. In the three dimensional case we can obtain similar upper and lower bounds. In the four-dimensional case the items time-dimension will be predefined and not altered, so all constraints concerning three dimensions applies also in this case. Since part of the problem now will be to determine the exact size of the items, this problem is non-linear. The item sizes are defined by non-linear and non-convex equalities,

$$h_i \cdot w_i \cdot l_i \cdot \ldots = R_i \quad i = 1, 2, \ldots, J \tag{4}$$

In equation (4), h_i, w_i and l_i are the height, width and length to be obtained (i.e. the elements of the vector $\mathbf{X}_i$) and R_i is the given size of item *i*. In the two-dimensional case R_i is the given area, in the three-dimensional case the given volume etc. Although equation (4) is a non-convex equality constraint it can generally, in the *N*-dimensional case be relaxed into convex inequality constraints (5). The convex inequality constraints for two- and three-dimensional cases are given as examples below.

$$h_i \cdot w_i \geq R_i \Rightarrow \begin{cases} f_1 : -h_i + \dfrac{R_i}{w_i} \leq 0 \\ f_2 : -w_i + \dfrac{R_i}{h_i} \leq 0 \end{cases}, h_i \cdot w_i \cdot l_i \geq R_i \Rightarrow \begin{cases} f_1 : -h_i + \dfrac{R_i}{l_i \cdot w_i} \leq 0 \\ f_2 : -w_i + \dfrac{R_i}{l_i \cdot h_i} \leq 0 \\ f_3 : -l_i + \dfrac{R_i}{w_i \cdot h_i} \leq 0 \end{cases} \quad i = 1,2,...,J \quad (5)$$

By linearization of (5) we obtain cutting-planes which can be added during optimisation of the problem. In the three dimensional case the cutting planes are, for example, defined by,

$$\left.\begin{aligned} -h_i - \frac{R_i}{\left(l_i^k\right)^2 \cdot w_i^k} \cdot l_i - \frac{R_i}{l_i^k \cdot \left(w_i^k\right)^2} \cdot w_i \leq -3\frac{R_i}{l_i^k \cdot w_i^k} \\ -w_i - \frac{R_i}{\left(l_i^k\right)^2 \cdot h_i^k} \cdot l_i - \frac{R_i}{l_i^k \cdot \left(h_i^k\right)^2} \cdot h_i \leq -3\frac{R_i}{l_i^k \cdot h_i^k} \\ -l_i - \frac{R_i}{\left(h_i^k\right)^2 \cdot w_i^k} \cdot h_i - \frac{R_i}{h_i^k \cdot \left(w_i^k\right)^2} \cdot w_i \leq -3\frac{R_i}{h_i^k \cdot w_i^k} \end{aligned}\right\} i = 1,2,...,J \quad (6)$$

Here l_i^k, h_i^k and w_i^k are the length, height and width obtained at iteration *k*. After subsequently adding cutting planes of the form (6) to a previous MILP problem, the problem is solved to global optimality using a simplification of the Extended Cutting Plane Method described in (Westerlund and Pörn, 2002).

4. Solving problems with MINDAT

All configuration and data files used by MINDAT are saved in plain ASCII text. The user can therefore define problem files with a text editor. Problems can however also be defined by a walkthrough wizard included in MINDAT. The problems can be solved with different options, such as for example any parameter configuration accepted by CPLEX. Figure 1 illustrates the output from MINDAT solving a single-floor process plant layout problem with 11 items. Problem data and previous results are found in (Westerlund, 2005, Paper VI, table 5-7). Best solution time for the problem achieved with MINDAT is 58 seconds on an AMD Athlon 64 3200+ computer with 1GB RAM.

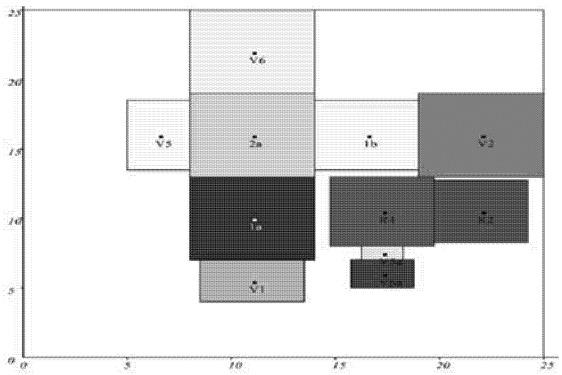

Figure 1: Output from MINDAT solving a single-floor process plant layout problem to optimum. The solution is not identical to the solution presented in (Westerlund, 2005) but is a multiple optimal solution.

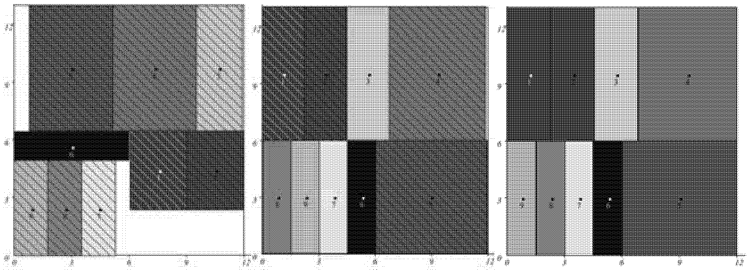

Figure 2: Output from MINDAT solving a block layout problem. Items marked with lines have still to reach their full valid sizes.

Figure 2 shows how a solution for a two-dimensional block layout problem evolves. When the optimisation starts, no cutting planes are yet added. This means that the items will be smaller than allowed after the first iterations. In MINDAT this is illustrated by lines on those items that are still too small. In the leftmost layout in figure 2, (obtained after two iterations, and adding cutting planes after the first iteration) we can see that only one item is large enough, and to the right we have the optimal allocation pattern with valid sized items. The layout in the middle of figure 2 is an intermediate solution. The problem data for the two dimensional block layout problem illustrated in figure 2 is the problem f09 in (Westerlund, 2005, p. 25) with aspect ratio 4. Previous results and solution data is found in (Castillo et. al. 2005). The best solution time with MINDAT for this two-dimensional, 9 item, block layout problem is 750 seconds on an AMD Athlon™ 64 3200+ computer with 1GB of RAM and CPLEX 10.0.1. In figure 3 MINDAT solves a 13 item, three-dimensional packing problem with four available containers. By clicking on a container in the three-view to the left, during or after optimisation, MINDAT draws the container with all its containing items.

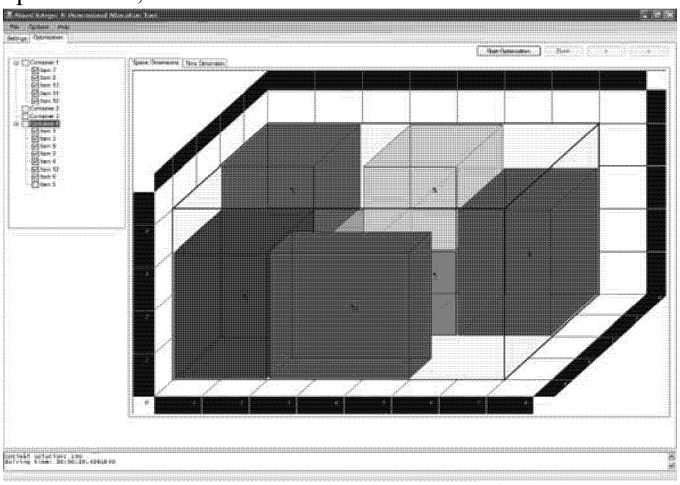

Figure 3: MINDAT solving a 3-dimensional packing problem to optimum. Items can be made transparent with a button-click to see what lies behind them. Problem data is found in (Westerlund, 2005, paper VI, table 8-9). The items in container 4 are illustrated in the figure.

5. References

Castillo, I. Westerlund, J. Emet, S. and Westerlund, T. 2005, Optimization of block layout design problems with unequal areas: A comparison of MILP and MINLP optimization methods. *Computers & chemical engineering* ***30*** , 54-69.

Westerlund, J. 2005, Aspects on *N*-dimensional allocation, PhD Thesis, Åbo Akademi University, Finland, ISBN 952-12-1625-5.

Westerlund, T. and Pörn, R. (2002), Solving Pseudo-Convex Mixed Integer Problems by Cutting Plane Techniques. *Optimization and Engineering,* ***3***, 253-280.