

A coevolutionary algorithm for a facility layout problem

T. DUNKER^{†*}, G. RADONS[‡] and E. WESTKÄMPER[†]

This paper presents a coevolutionary approach to the numerical optimization of large facility layouts. Our work is based on a mixed integer model for the layout constraints and objectives, which improves formulations found in the literature. Nevertheless, layouts with more than seven departments are difficult to solve. One way out is to apply genetic algorithms—searching systematically for solutions but without guarantee of finding an optimum. In this paper we suggest some improved mutation and cross-over operators. Yet, with increasing number of departments also genetic algorithms take very long. In this case we propose to use additional structures given by qualitative or quantitative reasoning. Clustering the departments into groups we allow each group ('species') to evolve (genetic algorithm) in a separate area while position and size of these areas ('environment') undergo an evolution, too. Numerical experiments verify this coevolutionary approach.

1. Introduction

One subproblem in factory planning consists in determining good locations of a set of departments (or manufacturing cells) on a planar site. This task is called the facility layout problem. The objectives briefly described by the word 'good' are manifold. In addition, many of them are of a qualitative nature and it is not straightforward to formulate them as measurable quantities. One objective is certainly to minimize the material handling cost. Yet, manpower requirements, work-in-process inventory, flow of information, etc. play an important role, too.

In all cases a basic assumption is that the proximity of certain departments is more favourable than other configurations. The aim is to arrange the departments in such a way that the desired proximity relations are satisfied.

There exist many simple examples. If all departments require only proximity with a predecessor and a successor a line layout is an adequate solution. A star-like layout might be a good design in a situation where few departments preprocess parts which are assembled in a final department. But, as soon as the number of departments increases and the exchange relations between the departments are more complex, one cannot fulfil all desired proximity relations and one has to decide between different possibilities. In such a case it is natural to apply computer programs for finding solutions.

For several decades there has been research on this subject. Meller and Gau (1996) present a detailed review of the different formulations of the facility layout

Revision received March 2003.

[†] Fraunhofer Institute of Manufacturing Engineering and Automation, Nobelstrasse 12, 70569, Stuttgart, Germany.

[‡] TU Chemnitz, Institut für Physik, Reichenhainer Strasse 70, 09126 Chemnitz, Germany.

* To whom correspondence should be addressed. email: tmd@ipa.fhg.de

problem and the variety of algorithms. Some additional recent references can be found in the introduction of Chiang (2001).

One can roughly distinguish between three types of problem formulation. Suppose there are a finite number of locations (e.g. on a lattice) and a set of departments or machines. In a first approach the task consists in assigning the departments to the different locations minimizing some cost function. This yields a quadratic assignment problem. A second approach starts with the available floor area and divides it successively into smaller subareas with respect to certain constraints which finally yields the locations for the departments. In a third formulation the necessary sizes and shapes of the departments are given and the departments can be placed arbitrarily within the available floor area. This last formulation, which results in a mixed integer programming (MIP) problem, forms the basis of our considerations in this paper.

All these approaches lead to combinatorial optimization problems which share the difficulty of a computational complexity growing very rapidly with the number of departments. Hence many suggested algorithms for treating these problems search heuristically for good solutions instead of aiming at globally optimal solutions. Evolutionary methods like genetic algorithms (GA) have successfully been applied in this context, see e.g. Chan and Tansri (1994), Rajasekharan *et al.* (1998), Azadivar and Wang (2000) or Tavares *et al.* (2000).

In this work we present a new evolutionary approach in order to attack such large-scale problems. Making use of additional structural features, which might be given by qualitative or quantitative reasoning, we apply principles of coevolution. Given a grouping of the set of departments we consider each group as a different 'species' and let them pass through an 'evolutionary process' in a common environment.

The remainder of this paper is organized as follows. In Section 2 we describe the mixed integer formulation of the facility layout problem chosen for our work. We present some improvements which reduce the number of binary variables and accelerate the solution. Section 3 provides the definition of the genetic operators and the description of the coevolutionary algorithm. We introduce new operators and demonstrate their excellent performance with some examples. Finally, in Section 4, we discuss the numerical results of our coevolutionary algorithm.

2. Mathematical formulation

2.1. Spatial considerations

2.1.1. Position and orientation

In order to make real facility layout problems manageable for computational algorithms a certain abstraction is necessary. The approximation of departments, floor areas, and others by rectangles is a popular method, see e.g. Rajasekharan *et al.* (1998) or Chiang (2001). In addition, we assume that their sides are parallel to the axis of our coordinate system in the plane. Furthermore, we suppose that the size and the shape of each department or facility is given in advance. Denoting a rectangle by $A \subset \mathbb{R}^2$ we introduce the following notation:

- $I = \{1, \dots, n\}$ index set of all rectangles,
- $i, j \in I$ indices for the rectangles,
- $l_i \in \mathbb{R}_+$ length of the long side of the i th rectangle,
- $s_i \in \mathbb{R}_+$ length of the short side of the i th rectangle,

- $(X_i, Y_i) \in \mathbb{R}^2$ coordinates of the centre point of the i th rectangle:
 $O_i \in \{0, 1\}$ orientation of the i th rectangle:
 $O_i = 1$ long side parallel to the x -axis,
 $O_i = 0$ long side parallel to the y -axis,
 $M_i^X \in \{0, 1\}$ flip up/down (symmetry axis parallel to x),
 $M_i^Y \in \{0, 1\}$ flip left/right (symmetry axis parallel to y).

Note that we use small letters for given constants, while potential variables—position and orientation—are denoted by capital letters. For a simple rectangle we need only two orientations but as soon as the internal structure of such an area is not symmetric with respect to its central axes we have to consider further transformations. Figure 1 shows all possible orientations of a rectangle with an internal structure.

Next, we wish to model nested areas. This is formalized below. Let i^* be the index of the total available floor area. We define a map $P: I \rightarrow I$, i.e. $i \mapsto P(i)$, with the following meaning. For each $i \in I$ the number $P(i)$ gives the index of a rectangle in which A_i shall be contained. For i^* we define $P(i^*) = i^*$. In this way we define a hierarchical structure of embedding relations. Most of the time one will have simply $P(i) = i^*$. Furthermore we will distinguish two types of rectangles—movable or fixed. For this purpose we define two index sets $I^m \subset I$ and $I^f \subset I$ with $I^m \cap I^f = \emptyset$ and $I^m \cup I^f = I$, where m stands for movable and f for fixed. Obviously, $i^* \in I^f$ and $i \in I^f$ will mean A_i is fixed to $A_{P(i)}$.

Depending on the containment structure P and the sets I^m and I^f one can determine whether the above-defined quantities will be variables or constants. Let us use the following notation:

$$P^{ok}(i) = \underbrace{P(\dots P(i) \dots)}_{k \text{ times}}.$$

Then we can distinguish the following three cases.

- (1) If $i \in I^m$ then certainly X_i , Y_i and O_i are variables. Whether there is a need for the variables M_i^X and M_i^Y depends on the internal structure of A_i , i.e. on the A_j 's with $P^{ok}(j) = i$ for some k .


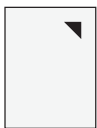
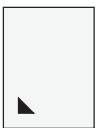
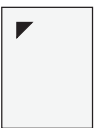




	$M_i^X = 0$ $M_i^Y = 0$	$M_i^X = 1$ $M_i^Y = 0$	$M_i^X = 0$ $M_i^Y = 1$	$M_i^X = 1$ $M_i^Y = 1$
$O = 0$				
$O = 1$				

Figure 1. All different orientations of a rectangle with an internal structure.

- (2) If $i \in I^f$ and $P^{ok}(i) \in I^f$ for all k then all quantities are constants and there is no need for M_i^X and M_i^Y .
- (3) If $i \in I^f$ and $P^{ok}(i) \in I^m$ for some k then set $k^* = \min\{k : P^{ok}(i) \in I^m\}$ and $j = P^{ok^*}(i)$. In this case X_i, Y_i and O_i are variables depending on the X_j, Y_j, O_j, M_j^X and M_j^Y . This will be investigated in the following paragraph.

In order to describe the transformation of a fixed rectangle A_i attached to a movable one A_j we need constants for modeling this dependency:

- $x_i^j, y_i^j \in \mathbb{R}$ relative coordinates of A_i with respect to A_j (long side corresponds to x-axis), see figure 2,
- $o_i^j \in \{0, 1\}$ relative orientation of A_i with respect to A_j (1 corresponds to parallel long sides), see figure 2.

Then we obtain the centre point coordinates from the relative coordinates by the linear transformation

$$\begin{pmatrix} X_i \\ Y_i \end{pmatrix} = \begin{pmatrix} X_j \\ Y_j \end{pmatrix} + \mathbf{M} \begin{pmatrix} x_i^j \\ y_i^j \end{pmatrix}, \tag{1}$$

with an orthonormal (2×2) matrix \mathbf{M} which can be decomposed in the following way

$$\mathbf{M} = \begin{pmatrix} U_j & 1 - M_j^Y & 1 \\ 1 - M_j^X & V_j & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 0 \end{pmatrix}.$$

This is more advantageous since the variables U_j and V_j take values in $\{0, 1\}$ only. One can verify that the constraints

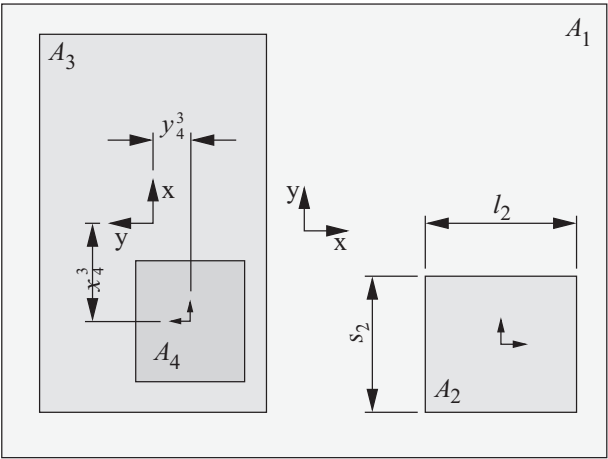


Figure 2. Illustration of the relative coordinates and orientations for nested rectangles. The relative orientation o_4^3 equals 1 as the long sides of A_4 and A_3 are parallel.

$$0 \leq O_j + M_j^Y - U_j \quad (2)$$

$$0 \leq O_j - M_j^Y + U_j \quad (3)$$

$$0 \leq -O_j + M_j^Y + U_j \quad (4)$$

$$O_j + M_j^Y + U_j \leq 2 \quad (5)$$

$$O_j + M_j^X - V_j \leq 1 \quad (6)$$

$$O_j - M_j^X + V_j \leq 1 \quad (7)$$

$$-O_j + M_j^X + V_j \leq 1 \quad (8)$$

$$1 \leq O_j + M_j^X + V_j \quad (9)$$

ensure the correct assignment of entries in the matrices \mathbf{M} . The orientation of A_i depends only on the orientation of the A_j and its relative orientation

$$O_i = \begin{cases} O_j & \text{for } o_i^j = 1 \\ 1 - O_j & \text{for } o_i^j = 0. \end{cases} \quad (10)$$

2.1.2. Geometric containment constraints

If $P(i) = j$ and $i \in I^m$ we can express the relation $A_i \subset A_j$ by the following four inequalities—for the left side

$$X_j - \frac{l_j}{2} O_j - \frac{s_j}{2} (1 - O_j) \leq X_i - \frac{l_i}{2} O_i - \frac{s_i}{2} (1 - O_i), \quad (11)$$

the right side

$$X_i + \frac{l_i}{2} O_i + \frac{s_i}{2} (1 - O_i) \leq X_j + \frac{l_j}{2} O_j + \frac{s_j}{2} (1 - O_j), \quad (12)$$

the bottom

$$Y_j - \frac{s_j}{2} O_j - \frac{l_j}{2} (1 - O_j) \leq Y_i - \frac{s_i}{2} O_i - \frac{l_i}{2} (1 - O_i) \quad (13)$$

and the top

$$Y_i + \frac{s_i}{2} O_i + \frac{l_i}{2} (1 - O_i) \leq Y_j + \frac{s_j}{2} O_j + \frac{l_j}{2} (1 - O_j). \quad (14)$$

Observe that as s_i , l_i , s_j and l_j are given constants the inequalities (11)–(14) are linear with respect to all other quantities.

2.1.3. Non-overlapping constraints

Now we turn to the part of the model which makes the numerical treatment so hard. The non-overlapping of two rectangles A_i and A_j means that $A_i \cap A_j = \emptyset$. In the literature there exist different suggestions for representing this placement restriction. The first mixed integer formulation of the facility layout problem which is due to Montreuil (1990) proposes four binary variables for each non-overlapping relation. In Meller *et al.* (1999) this approach is optimized in order to improve the performance of the branch and bound algorithm (b&b). Das (1993) and Rajasekharan *et al.* (1998) use a formulation which needs three binary variables.

We will propose a model using two variables only. Let us start with a short motivation by computational results.

We have compared our formulation to that used in Das (1993) and Rajasekharan *et al.* (1998) using the CPLEX MIP-solver (ILOG, Inc.) with two different branching strategies. The second problem from Das (1993) which consists of six departments served us as a test example. Table 1 shows that the number of nodes and the solution time (on a Pentium II 400 MHz) CPLEX needed for our formulation were considerably smaller.

Two non-overlapping rectangles A_i and A_j must be separated by a vertical or a horizontal line. We introduce variables

$$\begin{aligned} S_{ij}^D &\in \{0, 1\} && \text{line direction (vertical/horizontal)} \\ S_{ij}^O &\in \{0, 1\} && \text{order of } A_i \text{ and } A_j. \end{aligned}$$

The setting $(S_{ij}^D, S_{ij}^O) = (0, 0)$ stands for A_i left of A_j , $(0, 1)$ for A_i right of A_j , $(1, 0)$ for A_i below A_j and $(1, 1)$ for A_i above A_j , respectively. In order to avoid redundant variables we require $i > j$. Let $l_{\max} = \max_i l_i$. Using the four linear expressions

$$E_{ij}^{(1)} = l_{\max}(S_{ij}^D + S_{ij}^O) \tag{15}$$

$$E_{ij}^{(2)} = l_{\max}(1 + S_{ij}^D - S_{ij}^O) \tag{16}$$

$$E_{ij}^{(3)} = l_{\max}(1 - S_{ij}^D + S_{ij}^O) \tag{17}$$

$$E_{ij}^{(4)} = l_{\max}(2 - S_{ij}^D - S_{ij}^O) \tag{18}$$

where each takes the value zero for exactly one setting of (S_{ij}^D, S_{ij}^O) we can derive the necessary inequalities

$$X_i + \frac{l_i}{2}O_i + \frac{s_i}{2}(1 - O_i) \leq X_j - \frac{l_j}{2}O_j - \frac{s_j}{2}(1 - O_j) + E_{ij}^{(1)}, \tag{19}$$

$$X_j + \frac{l_j}{2}O_j + \frac{s_j}{2}(1 - O_j) \leq X_i - \frac{l_i}{2}O_i - \frac{s_i}{2}(1 - O_i) + E_{ij}^{(2)}, \tag{20}$$

$$Y_i + \frac{s_i}{2}O_i + \frac{l_i}{2}(1 - O_i) \leq Y_j - \frac{s_j}{2}O_j - \frac{l_j}{2}(1 - O_j) + E_{ij}^{(3)}, \tag{21}$$

$$Y_j + \frac{s_j}{2}O_j + \frac{l_j}{2}(1 - O_j) \leq Y_i - \frac{s_i}{2}O_i - \frac{l_i}{2}(1 - O_i) + E_{ij}^{(4)}. \tag{22}$$

Branching strategy	Automatic		Strong	
	Two	Three	Two	Three
Number of binaries				
Number of nodes ($\times 10^3$)	11.5	47.9	5	31
Time in s	25.5	187.6	46.3	512

Table 1. Comparison of the computational complexity (nodes of the b&b tree and time) of our model (two) with the model (three) which can be found, e.g. in Das (1993) or Rajasekharan *et al.* (1998). For the data of the test example with six departments see Das (1993).

There are relative positions which allow two settings of S_{ij}^D , e.g. if A_i is to the right of and above A_j so that they could be separated by a vertical as well as by a horizontal line. In order to break this symmetry we wish to require that if possible $S_{ij}^D = 1$ is chosen. This could be achieved by

$$Y_j - \frac{s_j}{2} O_j - \frac{l_j}{2} (1 - O_j) < Y_i + \frac{s_i}{2} O_i + \frac{l_i}{2} (1 - O_i) + l_{\max} S_{ij}^D \quad (23)$$

and

$$Y_i - \frac{s_i}{2} O_i - \frac{l_i}{2} (1 - O_i) < Y_j + \frac{s_j}{2} O_j + \frac{l_j}{2} (1 - O_j) + l_{\max} S_{ij}^D. \quad (24)$$

Unfortunately, these are strict inequalities which cannot be treated by a linear program. Using them with a ' \leq ' in our model we would miss only cases where A_i and A_j touch the same horizontal line from opposite sides.

Note that in our model for the genetic algorithm we will not make use of (23) and (24). On the contrary, we prefer to use the more flexible setting for the S_{ij}^D 's and S_{ij}^O 's.

If the problem is invariant with respect to rotation and reflection then we can break these symmetries by fixing, e.g. $S_{21}^D = 1$ (rotation), $S_{21}^O = 1$ (reflection symmetry along x) and by adding the constraint $0 \leq X_2 - X_1$ (reflection symmetry along y).

Finally, we denote by $I^{\text{no}} \subset \{(i, j) : i > j; i, j \in I\}$ the set of the non-overlapping relations which are necessary for the model. The aim is to keep I^{no} as small as possible. For example consider the following situation: $A_2 \subset A_1 = A_r$, $A_3 \subset A_1$, $A_4 \subset A_2$, $A_5 \subset A_2$, $A_6 \subset A_3$ and $A_7 \subset A_3$. In order to have the no overlap between A_2 , A_3 and A_4, \dots, A_7 it suffices to set $I^{\text{no}} = \{(3, 2), (5, 4), (7, 6)\}$. The relations $(6, 4), (6, 5), (7, 4)$ and $(7, 5)$ are automatically true because of the containment relation and $(3, 2)$.

2.2. Distance based objective function

The objective is to minimize the distances between different points. In most cases these points will represent the drop off and pick up points of a department. However, we can also imagine other points which possess a certain importance, e.g. for the production process, the internal communication process, or the security. In the following we will call these points IO-points. They are attached to some floor area or a department, i.e. a rectangle may possess no, one, two or more IO-points. Hence, whether they are fixed or how they can be moved, depends on the corresponding rectangles. We use the following notation:

$I^* = \{1, \dots, n^*\}$	set of the indices of the IO-points,
$\alpha, \beta \in I^*$	indices for the IO-points,
$(x_\alpha^*, y_\alpha^*) \in \mathbb{R}^2$	relative coordinates of the α -th IO-point, $\alpha \in I^*$, with respect to the centre of the i th rectangle, $i \in I^m$,
$(X_\alpha^*, Y_\alpha^*) \in \mathbb{R}^2$	coordinates of the α th IO-point which are either constant (IO-point is attached to A_r) or variable satisfying e.g. equation (1) with the corresponding relative coordinates,
$\delta X_{\alpha\beta}^*, \delta Y_{\alpha\beta}^* \in \mathbb{R}_+$	horizontal and vertical distance between the α th and β th IO-point,
$(w_{\alpha\beta})_{\substack{\alpha, \beta \in I^* \\ \alpha > \beta}}$	weights for the distances representing e.g. material handling cost per meter.

Given an existing layout it might be necessary to consider the cost of changing the layout at the same time. For this purpose we introduce for all $i \in I^m$:

- $X_i^{(0)}, Y_i^{(0)}, O_i^{(0)}$ position and orientation values of the original layout,
 $M_i \in \{0, 1\}$ this variable is 1 if X_i, Y_i or O_i have changed from the original layout or if M_i^X or M_i^Y is equal to 1,
 $(w_i)_{i \in I^m}$ weights for moving a rectangle representing e.g. the cost for moving a department.

With this notation the objective is to minimize the following expression

$$\min \left(\sum_{\substack{\alpha, \beta \in I^* \\ \alpha > \beta}} w_{\alpha\beta} (\delta X_{\alpha\beta}^* + \delta Y_{\alpha\beta}^*) + \sum_{i \in I^m} w_i M_i \right) \quad (25)$$

where the following constraints have to be satisfied

$$X_{\alpha}^* - X_{\beta}^* \leq \delta X_{\alpha\beta}^* \quad (26)$$

$$X_{\beta}^* - X_{\alpha}^* \leq \delta X_{\alpha\beta}^* \quad (27)$$

$$Y_{\alpha}^* - Y_{\beta}^* \leq \delta Y_{\alpha\beta}^* \quad (28)$$

$$Y_{\beta}^* - Y_{\alpha}^* \leq \delta Y_{\alpha\beta}^* \quad (29)$$

for all $\alpha, \beta \in I^*$ with $\alpha > \beta$ and $w_{\alpha\beta} \neq 0$ and

$$X_i - X_i^{(0)} \leq l_{\max} M_i \quad (30)$$

$$X_i^{(0)} - X_i \leq l_{\max} M_i \quad (31)$$

$$Y_i - Y_i^{(0)} \leq l_{\max} M_i \quad (32)$$

$$Y_i^{(0)} - Y_i \leq l_{\max} M_i \quad (33)$$

$$O_i - O_i^{(0)} \leq M_i \quad (34)$$

$$O_i^{(0)} - O_i \leq M_i \quad (35)$$

$$M_i^X \leq M_i \quad (36)$$

$$M_i^Y \leq M_i \quad (37)$$

for all $i \in I^m$.

If all objects attached to a rectangle A_i are located on the same symmetry axis then it is not necessary to employ (1) with all the constraints (2)–(9). In this case we need only two binary variables $U_i, V_i \in \{0, 1\}$ implementing the rotation and a single reflection. Consider, e.g. a IO-point α attached to the rectangle A_i . We can use the following model with inequality constraints

$$-2O_i \leq U_i - V_i \leq 2O_i \quad (38)$$

$$O_i \leq U_i + V_i \leq 2 - O_i \quad (39)$$

and equations

$$X_{\alpha}^* = X_i + x_{\alpha}^{*i} (U_i + V_i - 1) \quad (40)$$

$$Y_{\alpha}^* = Y_i + x_{\alpha}^{*i} (U_i - V_i) \quad (41)$$

for the symmetry axis parallel to the long side or, alternatively,

$$X_{\alpha}^{\bullet} = X_j + y_{\alpha}^{\bullet i}(U_i - V_i) \quad (42)$$

$$Y_{\alpha}^{\bullet} = Y_j + y_{\alpha}^{\bullet i}(U_i + V_i - 1) \quad (43)$$

for the symmetry axis parallel to the short side of the rectangle. Thus our mixed integer programming model is completely described by the equations (1)–(43). All equalities, inequalities and the objective are linear. The model contains binary variables O_i , M_i^X , M_i^Y , U_i , V_i , M_i , S_{ij}^D and S_{ij}^O . While the number of S_{ij}^D 's and S_{ij}^O 's increases quadratically, the number of the others increases linearly with the number of rectangles.

3. Coevolutionary algorithm

The goal of finding an optimal solution of the mathematical model introduced in Section 2 and to prove its optimality can be achieved only for a small number of departments (up to ≈ 7). This is due to the quadratic increase in the number of binary variables. That is why various heuristic methods have been developed to find systematically at least suboptimal solutions.

One such approach uses genetic algorithms, see e.g. Chan and Tansri (1994), Conway and Venkataramanan (1994), Gero and Kazakov (1998), Kochhar and Heragu (1998), Kochhar and Heragu (1998) (all quadratic assignment problems), Shnecke and Vornberger (1997), Azadivar and Wang (2000) (both slicing tree representations), Rajasekharan *et al.* (1998), Tavares *et al.* (2000) (both mixed integer models), Gau and Meller (1999) (slicing tree and mixed integer). In our case the information about the setting of the binary variables is translated into a genetic code. Then a population of individuals carrying this genetic code undergoes an evolutionary process which creates improved generations of this population by selection, cross-over, and mutation.

In the following we introduce a coevolutionary approach which goes beyond the standard genetic algorithms. The philosophy of coevolution can be described as follows. Let us suppose that a large problem can be decomposed into smaller ones which are linked to each other. Then one can assign to each such subproblem a population of individuals representing possible solutions. Different subproblems form different species which undergo an evolution. Observe that there is no exchange of genetic material between different species. However, the fitness of an individual from one population now depends also on the other populations.

One interesting field of research is to obtain the different species themselves by an evolutionary process of specialization. In our case we generate the problem decomposition by ourselves. We form groups of departments. For each group we reserve a separate area. Inside each such area group layouts are evolved by genetic algorithms. The fitness of one group layout depends in addition on the best layouts of the other groups. This is the coevolutionary part of our algorithm. A second genetic algorithm changes size and position of the areas. This is done for two purposes. First this allows further improvement. Secondly, by changing the size we can control the evolution of a group—more space allows more variation while tightening up stops evolution.

The remainder of this section is organized as follows. First, we describe the genetic operators which are adapted to the problem. Then the coevolutionary algorithm is described in detail.

3.1. An adapted genetic algorithm

In Rajasekharan *et al.* (1998) a (0–1) sequence for setting the binary variables is used as the genetic code. Standard cross-over and mutation operators are applied. Obviously, some of the outcomes are infeasible settings. If an infeasible gene occurs it is eliminated from the population immediately. There are for example, binary variables which represent the order of the x -coordinate of two centre points. If we take six centre points then we have 15 such variables giving 32768 possible genes, but there are just $6! = 720$ possible arrangements, e.g. many of the settings are infeasible as they do not satisfy the transitivity of relative positions (if A_i is left of A_j and A_j is left of A_k then A_i cannot be right of A_k , it has to be left of A_k).

In order to avoid producing too many infeasible genes we introduce a coding and operators which do not leave the space of feasible settings.

3.1.1. Coding

We assume that the problem to solve involves the rectangles with indices from the index set $I_k^g \subset I$ where the letter g stands for ‘group’ and $k = 1, 2, \dots$ is the index of the group. Let $N_k^g = |I_k^g|$ be the number of relevant rectangles. Denote by $\iota = 1, \dots, n_k^p$ the index of an individual within a population of n_k^p individuals. Let $\gamma = 1, 2, \dots$ be the index of the generation. The ι th individual of the k th group in generation γ will be represented by

$$\mathbb{I}_{k,\iota}^{(\gamma)} = \left((i_1^x, \dots, i_{n_k^g}^x), (i_1^y, \dots, i_{n_k^g}^y), \{b_{ij}\}_{i,j \in I_k^g, i>j} \right).$$

The $b_{ij} \in \{0, 1\}$, with $i, j \in I_k^g$ and $i > j$, represent values of the binary variable S_{ij}^D , i.e. whether there is a vertical or horizontal separating line between the rectangles A_i and A_j . The two vectors $(i_1^x, \dots, i_{n_k^g}^x)$ and $(i_1^y, \dots, i_{n_k^g}^y)$ are permutations of the elements of I_k^g and they represent the order of the x - and y -coordinates of the midpoints of the rectangles.

Next we describe the translation of this genetic code to the variables of the model described in section 2. Given an $\mathbb{I}_{k,\iota}^{(\gamma)}$ we set the S_{ij}^D and S_{ij}^O with $i, j \in I_k^g$ and $i > j$ in the following way. For each pair of indices $1 \leq j_1 < j_2 \leq n_k^g$ we check the following alternatives. Set $i^x = \max(i_{j_1}^x, i_{j_2}^x)$ and $j^x = \min(i_{j_1}^x, i_{j_2}^x)$. If $b_{i^x j^x} = 0$ holds, i.e. there is a vertical separating line between A_{i^x} and A_{j^x} , then set

$$S_{i^x j^x}^D = 0 \tag{44}$$

$$S_{i^x j^x}^O = \begin{cases} 0 & \text{if } i^x = i_{j_1}^x \\ 1 & \text{if } i^x = i_{j_2}^x. \end{cases} \tag{45}$$

Note that if $b_{i^x j^x} = 1$ the order $i_{j_1}^x, i_{j_2}^x$ does not enter in the problem formulation. Hence the final order of the x -coordinates of $A_{i_{j_1}^x}$ and $A_{i_{j_2}^x}$ is not necessarily $X_{i_{j_1}^x} \leq X_{i_{j_2}^x}$. Analogously, for the y -direction we set $i^y = \max(i_{j_1}^y, i_{j_2}^y)$ and $j^y = \min(i_{j_1}^y, i_{j_2}^y)$. If $b_{i^y j^y} = 1$ is true, i.e. there is a horizontal separating line between A_{i^y} and A_{j^y} , then we assign

$$S_{i^y j^y}^D = 1 \quad (46)$$

$$S_{i^y j^y}^O = \begin{cases} 0 & \text{if } i^y = i_{j_1}^y \\ 1 & \text{if } i^y = i_{j_2}^y. \end{cases} \quad (47)$$

3.1.2. Genetic operators

Chan and Tansri (1994) introduced three different cross-over operators for permutations—partially matched, order and cycle cross-over. For our genetic algorithm we use a version of the order cross-over. After selecting two parent genes $\mathbb{I}_{k,l}^{(\gamma)}$ and $\mathbb{I}_{k,l_2}^{(\gamma)}$ let us consider the parts of the genes representing the x - and the y -order. Take, for example,

$$(i_{1,l_1}^x, \dots, i_{n_k^g}^x, l_1) \quad \text{and} \quad (i_{1,l_2}^x, \dots, i_{n_k^g}^x, l_2)$$

where we add the number of the individual as a second subindex. We select randomly two cut positions $c_1, c_2 \in \{1, \dots, n_k^g\}$ with $c_1 \leq c_2$. Then we construct two new genes from the two selected genes. First we fill the position from c_1 to c_2 with the original parts of the sequence

$$(\dots, i_{c_1,l_1}^x, \dots, i_{c_2,l_1}^x, \dots) \quad \text{and} \quad (\dots, i_{c_1,l_2}^x, \dots, i_{c_2,l_2}^x, \dots).$$

Then the positions to the left of c_1 and to the right of c_2 are filled with the numbers from the other parent which are not contained in the already filled part. While filling we keep the order given by the parent we take the elements from. In terms of the notation above this means, e.g. for the first offspring gene

$$(i_{f(1),l_2}^x, \dots, i_{f(c_1-1),l_2}^x, i_{c_1,l_1}^x, \dots, i_{c_2,l_1}^x, i_{f(c_1),l_2}^x, \dots, i_{f(n_k^g-c_2+c_1-1),l_2}^x)$$

with

$$\{i_{f(1),l_2}^x, \dots, i_{f(n_k^g-c_2+c_1-1),l_2}^x\} \cap \{i_{c_1,l_1}^x, \dots, i_{c_2,l_1}^x\} = \emptyset$$

and the mapping $j \mapsto f(j)$ is strictly increasing. In the same way the cross-over is defined for the y -direction. The motivation for this definition is the idea that we wish to keep the part that is located between the cuts, hoping that it contributes to a good solution and arranging the remaining elements in the order given by the other parent.

For mutation a single parent is randomly selected. Then the mutation operator just exchanges two randomly chosen elements in $(i_1^x, \dots, i_{n_k^g}^x)$ and independently also in $(i_1^y, \dots, i_{n_k^g}^y)$.

The more complicated part is the modification of $\{b_{ij}\}_{i,j \in I_k^g, i > j}$ which codes the decision of having a vertical or horizontal separating line. As we did not find a method which could be geometrically motivated we decided to use the standard cross-over with two cut positions and the standard mutation. However, in addition we apply an improvement strategy. First, we fix the variables S_{ij}^D and S_{ij}^O for the given individual $\mathbb{I}_{k,l}^{(\gamma)}$ according to (44)–(47) and solve the remaining mixed integer problem (1)–(43):

```

evaluate  $(\mathbb{I}_{k,l}^{(\gamma)})$ 
  for all  $i, j \in I_k^g$  with  $i > j$ 
    fix  $S_{ij}^D$  and  $S_{ij}^O$ 
  endfor
  solve remaining mixed integer problem
  return solution

```

Next, we update $(i_1^x, \dots, i_{n_k^g}^x)$ and $(i_1^y, \dots, i_{n_k^g}^y)$ by sorting the centre point coordinates of the obtained solution. Then we check for all S_{ij}^D whether they can be changed without violating a constraint in the current solution.

```

change_is_possible ( $S_{ij}^D$ )
  if  $S_{ij}^D = 0$  (x-direction)
    if  $|Y_i - Y_j| \geq s_i O_i / 2 + l_i(1 - O_i) / 2 + s_j O_j / 2 + l_j(1 - O_j) / 2$ 
      return true
    else
      return false
    endif
  else (y-direction)
    if  $|X_i - X_j| \geq l_i O_i / 2 + s_i(1 - O_i) / 2 + l_j O_j / 2 + s_j(1 - O_j) / 2$ 
      return true
    else
      return false
    endif
  endif

```

If it is possible we change the variable S_{ij}^D . These actions are repeated until the objective value does not decrease further. Summarizing, we have sketched our improvement strategy below:

```

improve ( $\mathbb{I}_{k,l}^{(\gamma)}$ )
  while the objective value decreases
    evaluate ( $\mathbb{I}_{k,l}^{(\gamma)}$ )
    obtain  $(i_1^x, \dots, i_{n_k^g}^x)$  and  $(i_1^y, \dots, i_{n_k^g}^y)$  from the solution
    for all  $i, j \in I_k^g$  with  $i > j$ 
      if change_is_possible ( $S_{ij}^D$ )
        if  $S_{ij}^D = 0$ 
           $b_{ij} = 1$ 
        else
           $b_{ij} = 0$ 
        endif
      endif
    endfor
  endwhile
  return  $\mathbb{I}_{k,l}^g$  with smallest objective value

```

We have tested these operators with a standard genetic algorithm with a population of 50 different individuals. Each new generation is created by $n^{\text{cr}} = 20$ cross-over operations, $n^{\text{mu}} = 5$ mutations and copying the $n^{\text{co}} = 5$ best individuals. The selection for the cross-over and the mutation accepts individuals with objective value above average of the population with a probability of $p^{\text{ac}} = 20\%$. The mutation rate is chosen to be $r^{\text{mu}} = 10\%$. We terminate if the average of the objective values has not changed more than $m^{\text{ch}} = 0.01\%$, or the average of the best values has not changed for the last $n^{\text{nc}} = 10$ generations, or a maximal number $m^{\text{ge}} = 1000$ of generations has been exceeded.

genetic_algorithm**initialize** population and compute the average of the objective values**do** $\gamma = \gamma + 1$ **for** $1, \dots, n^{\text{cr}}$ **select** parents from generation $\gamma - 1$ **cross-over** generates two new individuals for generation γ **endfor****for** $1, \dots, n^{\text{mu}}$ **select** parent from generation $\gamma - 1$ **mutation** generates a new individual for generation γ **endfor****copy** the n^{co} best individuals from generation $\gamma - 1$ to generation γ
compute the average of the objective values and determine whether
the best individual has changed**while** the change of the average is larger than m^{ch} and the best individual
has changed during the last n^{nc} generations and $\gamma \leq m^{\text{ge}}$

Applying this to the third example from Das (1993) with eight departments we obtained satisfactory results. Running the deterministic algorithm (31 h 30 min on a Pentium III 866 MHz with a memory use of approx. 1.5 GB) it has been proved that the optimal objective value is 8778.3. Running our genetic algorithm 13 times, the optimal objective value was reached three times. In the worst case the objective value was 9106.6 which lies just 3.7% above the optimum. Figures 3 and 4 show the convergence and distribution of the solution values, respectively. In all cases computations took less than 10 minutes (on a Pentium II 400 MHz). Also for all other examples our approach shows very good results. Table 2 compares our best results to

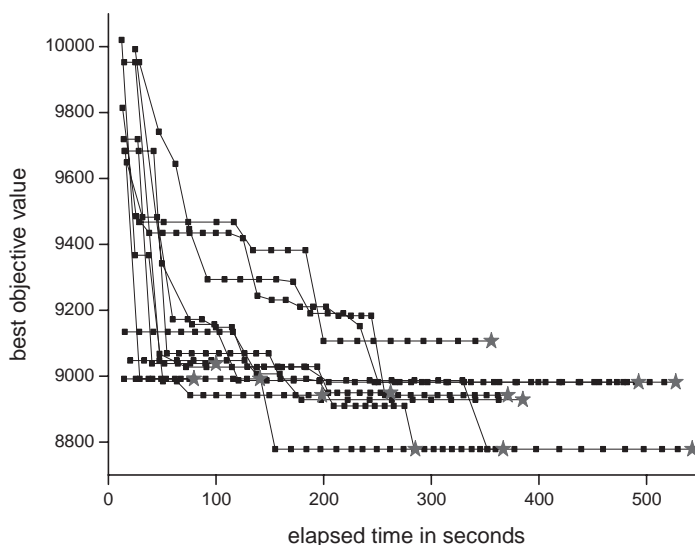


Figure 3. Convergence of the best objective values.

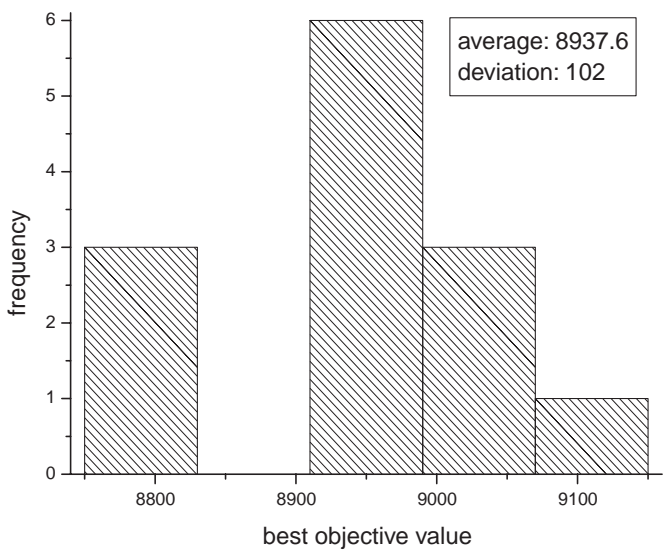


Figure 4. Distribution of the objective values.

the best known from Das (1993) and Rajasekharan *et al.* (1998) showing a considerable improvement in all cases.

3.2. Description of the coevolutionary algorithm

If we attack large problems with the above-described simple genetic algorithm we have to wait a very long time for good results. One way to obtain results more rapidly could be stopping the algorithm if it exceeds a time limit. The result might be good. However, the quality of this method would vary more or less randomly.

One may question whether large problems are really that large in practice. Most of the time we have additional information which can reduce the complexity considerably. By some quantitative or qualitative method we can form groups of departments which should be placed together. Then the problem reduces to two smaller problems (in terms of the number of objects to be placed). One has to provide an area for each group of departments and one has to determine the layout for each group within these areas. Tam and Li (1991) suggested approaching the facility layout problem in a hierarchical manner by a divide-and-conquer strategy. They formed groups, computed the layout for each of them, and placed the groups in a final step.

Number of departments	Four-step method described in Das (1993)	Genetic algorithm by Rajasekharan <i>et al.</i> (1998)	Best results of our genetic algorithm as described above
8	10 777.1	9 174.8	8 778.3
10	15 878.3	19 777.3	15 694.5
12	41 267.5	45 353.5	37 396.1

Table 2. Minimal objective value for three example problems from Das (1993) which are reported in Das (1993) and Rajasekharan *et al.* (1998) in comparison with our best results.

We propose a coevolutionary method of iterative nature. In a first step we find initial layouts for each group. Next, we fit a rectangle around each group and enlarge each side by a factor $1 + z_k$ giving more space to each group for possible further change. This allows, e.g. that a group becomes more oblong during the subsequent optimization. Next, we arrange these rectangles using again a genetic algorithm. In the first run we approximate the IO-points by the central points of the rectangles. Afterwards it is necessary to consider all relative positions of the IO-points of the group. Experiments with continued approximation by the central point did not show satisfactory results.

Keeping the external IO-points for all groups constant, each group undergoes a short evolution by a genetic algorithm. Observe that the objective function does not only include the weighted distances between the group members; the weighted distances to the constant IO-points outside the group contribute to the objective function, too. It is obvious that these genetic algorithms can be computed in parallel. For each group we decide whether we change z_k for the next iteration. For this purpose we compare the new dimensions to the old ones. If the proportional increase

$$\max\left(\frac{\max(l_{\text{new}} - l_{\text{old}}, 0)}{l_{\text{old}}}, \frac{\max(s_{\text{new}} - s_{\text{old}}, 0)}{s_{\text{old}}}\right)$$

exceeds a certain percentage p^+ of z_k then z_k is multiplied by a factor $f > 1$. If it remains below $p^- z_k$ then z_k is divided by the same factor f . Consequently, if the shape of the needed area does not change the provided group area becomes tighter. We stop the iteration when all group areas are close to the needed area of the group. The algorithm is summarized below.

coevolutionary_algorithm

for all groups k

genetic_algorithm find a good layout for the departments of group k (open floor)

fit a group area around the group and enlarge it by z_k (i.e. multiply the length of each side by the factor $1 + z_k$)

endfor

do

genetic_algorithm find a good layout for the group areas (consider each area as a department with several IO-points)

for all groups k *** *coevolution* ***

genetic_algorithm find a good layout for the departments of group k (the placement is restricted to the group area and all external IO-points are fixed) fit a new group area around the group

if the ratio of the sides has changed much

increase z_k

else

decrease z_k

endif

enlarge the new group area by z_k

endfor

while there is still a 'large' z_k for some k (or the average of the z_k 's is large)

Let us add a remark. When the provided floor area for a group becomes smaller and smaller it is more difficult for the genetic algorithm to find new feasible layouts

for this group. An awkward choice of the S_{ij}^D and S_{ij}^O may prevent the rectangles from fitting inside the prescribed area. However, as we keep the genetic pool from the last iteration we can copy at least one feasible individual to the population of the next generation.

4. Results and conclusions

For our numerical experiments we created a random example with 62 departments (rectangles) of different shapes. For simplicity we placed one IO-point in the centre of each department. The weights $w_{\alpha\beta}$ were generated randomly, too. In this example, we do not start from an initial layout. Hence, all the weights w_i in (25) are zero. Table 3 in the appendix provides all the necessary quantities. Let us call this problem P62.

In order to find a grouping we implemented the heuristic grouping algorithm of Harhalakis *et al.* (1990). A similar clustering algorithm was applied by Tam and Li (1991). The aim is to arrange the departments in groups minimizing the sum of the weights between departments belonging to different groups. In addition, one limits the size of each group. In De Lit *et al.* (2000) one can find grouping algorithms using genetic algorithm. One can construct examples where the deterministic heuristic by Harhalakis *et al.* (1990) stops far from the optimum as it can handle only simple exchange operations. In these cases algorithms like the one by De Lit *et al.* (2000) can improve the grouping.

For our tests we generated groupings with four, six, eight and nine groups with a maximal group size of 16, 11, eight and seven departments, respectively. Minimizing the sum of weights w_{ij} between IO-points from different groups is equivalent to maximizing the sum of weights connecting departments of the same group. All the groupings we used are summarized in table 4 in the appendix.

There are several parameters which influence the performance of the proposed coevolutionary algorithm. For the parameters introduced in section 3.1 and 3.2 we used the settings $r^{\text{mu}} = 10\%$, $p^{\text{ac}} = 20\%$, $n^{\text{nc}} = 5$, $m^{\text{ch}} = 0.5\%$; the values in the following table:

	n^{cr}	n^{mu}	n^{co}	m^{ge}
First arrangement of the areas for each group	20	5	5	15
Adjusting the positions of the changed areas	0	5	5	1
Coevolution of the ‘group’ populations	12	3	3	3

and initial enlargement factor $z_k = 0.6$, increase limit $p^+ = 75\%$, decrease limit $p^- = 50\%$ and change factor $f = 1.5$.

There is always a conflict between good exploration of the search space and fast computation. The first requires large populations which again needs more time for computation. Certainly, these parameters can still be optimized.

We applied the coevolutionary algorithm about 20 times to each grouping of P62. Figure 5 shows the convergence behaviour of the different runs and, in addition, displays the average, standard deviation, and the best and worst case of both the objective values of the solutions and the time of the computations. In figure 8, in the appendix, one can find the best layouts obtained. We observe that both the average objective value and the time of computation increase with the number of

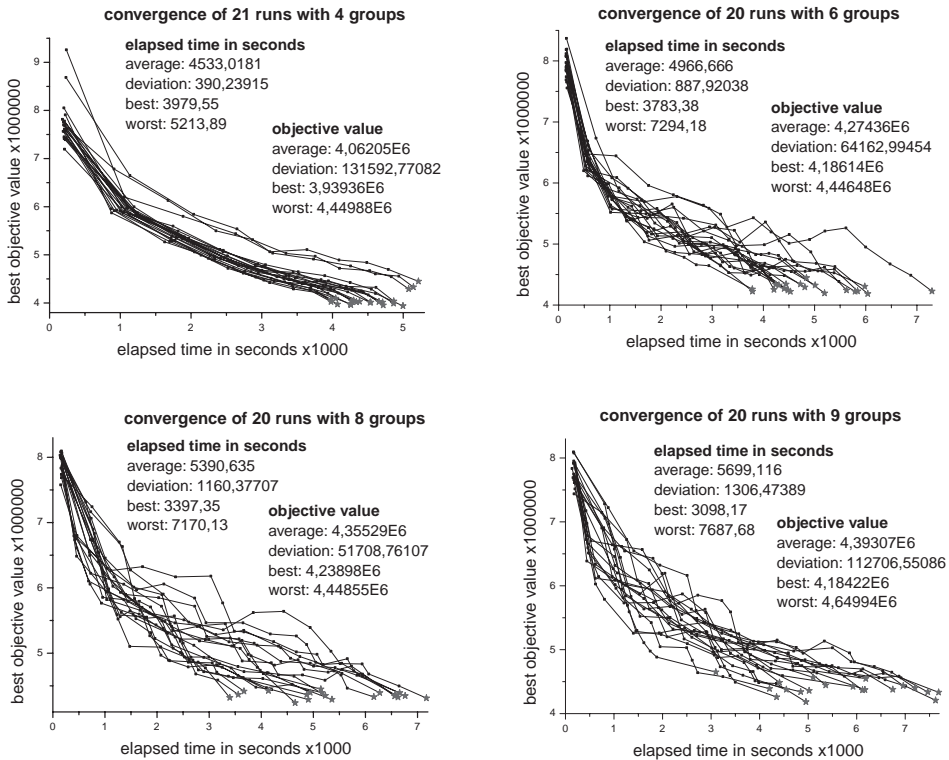


Figure 5. Computational results for problem P62 with four, six, eight and nine groups (obtained on a PC, Pentium IV, 1.5 GHz).

groups. The first is due to a worse packing. Since the groups never exactly fill the provided rectangular area there is more space lost when the number of groups increases. Secondly, it turned out that the computations treating the part where whole groups are moved are very time consuming. Here not only the orientation also the different reflection symmetries have to be considered. This is the reason why the time of computation increases. Thus the clustering into four groups appears to be the best choice for a facility layout problem of this size—the restrictions introduced by the grouping are the least yet the computation is still fast.

In contrast to computations of less than two hours of the coevolutionary algorithm the simple genetic algorithm needs on average about two days and 17 hours for P62. The quality of the solution (objective value) lies in the same range as the solutions obtained by the coevolutionary algorithm. For comparison the corresponding results for 11 runs on a Pentium IV, 1.5 GHz are summarized in figure 6. A trial with a MIP-solver, a good starting solution and lower bounds did not yield any feasible solution after one week.

Of course, there are further related problems of interest, which we will not treat here. For example, one may ask what happens when the problem size is further increased. Is there a point from which on six groups perform better than four? A second interesting question is whether dividing groups into subgroups can be of advantage in some cases. Here one introduces further restrictions so one would in general expect worse objective values.

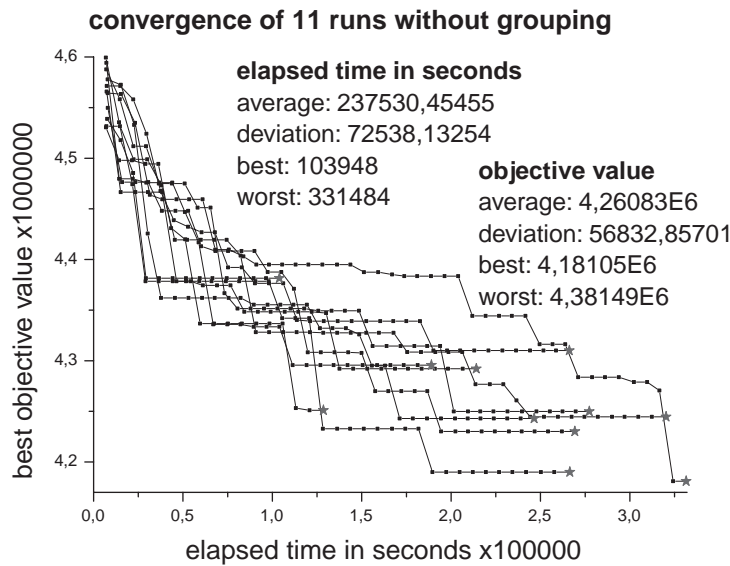


Figure 6. Summary of computational results of the genetic algorithm without grouping for P62. While the values of the objective function are in the same range as the results obtained with grouping, the computation time exceeds that of the coevolutionary algorithm by an order of magnitude.

Summarizing, we conclude that the proposed coevolutionary algorithm opens the possibility of finding good solutions for large facility layout problems within some hours where global optimization algorithms fail. In addition there is still a high potential for further computational acceleration by parallelization, if appropriate hardware is available.

Acknowledgements

We would like to thank the referees for their valuable suggestions. This research was supported by DFG research project SFB 467 ‘Wandlungsfähige Unternehmensstrukturen für die variantenreiche Serienproduktion’.

Appendix: Solutions and data

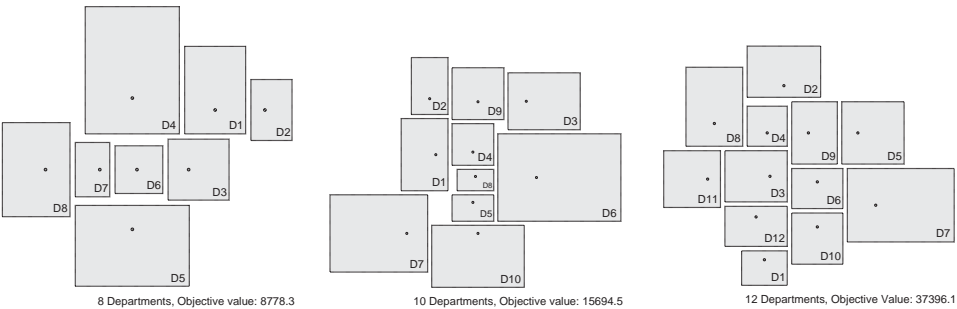
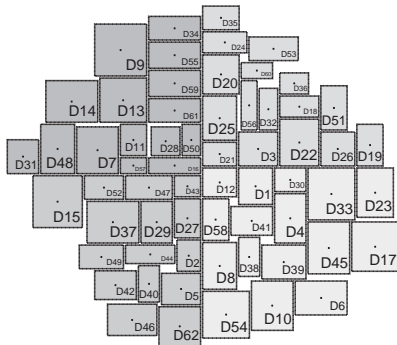


Figure 7. Best layouts for the last three example problems from Das (1993) found by the above-described GA. For the example with eight departments it was possible to prove optimality by solving the complete mixed integer problem.

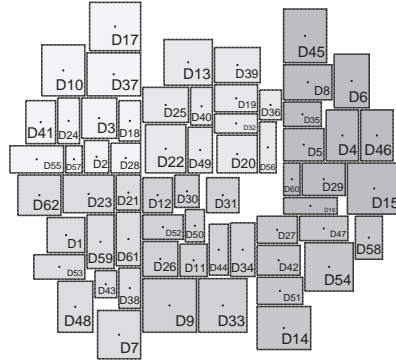
Table 3. Side lengths and weight matrix for the randomly generated example of 62 departments (P62).

Number	Group elements	$\sum_{\text{group}} w_{\alpha\beta}$
1	{1, 4, 6, 8, 10, 12, 17, 23, 30, 33, 38, 39, 41, 45, 54, 58}	4819
2	{3, 18, 19, 20, 21, 22, 24, 25, 26, 32, 35, 36, 51, 53, 56, 60}	4998
3	{2, 5, 15, 27, 29, 37, 40, 42, 43, 44, 46, 47, 49, 52, 62}	4944
4	{7, 9, 11, 13, 14, 16, 28, 31, 34, 48, 50, 55, 57, 59, 61}	4417
Sum:		19 178
1	{2, 3, 10, 17, 18, 24, 28, 37, 41, 55, 57}	2245
2	{13, 19, 20, 22, 25, 32, 36, 37, 40, 49, 56}	2632
3	{1, 7, 21, 23, 38, 43, 48, 53, 59, 61, 62}	2879
4	{9, 11, 12, 26, 30, 31, 33, 34, 44, 50, 52}	2375
5	{14, 27, 42, 47, 51, 54, 58}	671
6	{4, 5, 6, 8, 15, 16, 29, 35, 45, 46, 60}	2649
Sum:		13 451
1	{1, 12, 23, 33, 38, 45, 54, 58}	1605
2	{18, 19, 20, 24, 25, 26, 32, 36}	1222
3	{27, 37, 40, 42, 46, 52, 62}	946
4	{3, 21, 22, 35, 51, 53, 56, 60}	1528
5	{6, 7, 10, 11, 13, 28, 48, 57}	1244
6	{4, 8, 16, 30, 31, 39, 41, 50}	1380
7	{9, 14, 17, 34, 55, 59, 61}	1165
8	{2, 5, 15, 29, 43, 44, 47, 49}	1699
Sum:		10 789
1	{7, 21, 22, 35, 48, 53, 60}	1067
2	{1, 23, 38, 43, 54, 61, 62}	1462
3	{4, 6, 18, 24, 45, 55, 59}	1176
4	{10, 13, 19, 28, 36, 42, 56}	1274
5	{2, 3, 5, 17, 33, 41, 51}	909
6	{16, 20, 25, 32, 39, 40, 49}	1263
7	{11, 15, 27, 29, 46, 47, 52}	1324
8	{9, 12, 26, 30, 31, 50}	811
9	{8, 14, 34, 37, 44, 57, 58}	849
Sum:		10 135

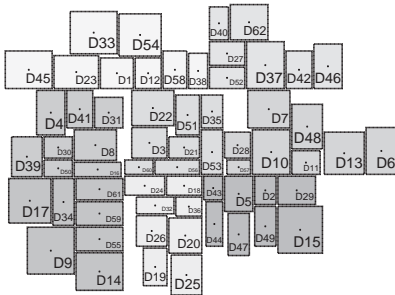
Table 4. Grouping all departments into four, six, eight and nine groups, respectively.



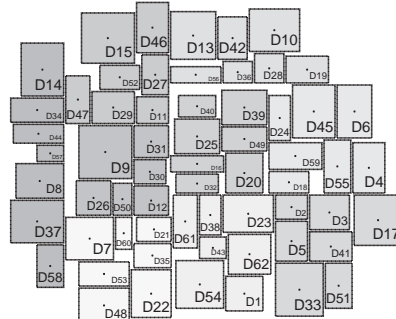
Result, Iter. 9, Total Cost: 3939362.0 (4996 sec.)



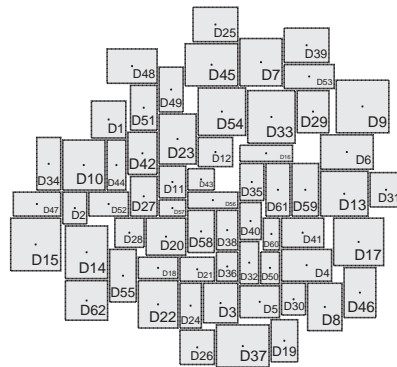
Result, Iter. 13, Total Cost: 4186140.4 (6038 sec.)



Result, Iter. 10, Total Cost: 4238981.6 (4651 sec.)



Result, Iter. 11, Total Cost: 4184224.0 (4955 sec.)



62 departments, objective function: 4181054.0

Figure 8. The best layouts for four, six, eight and nine groups obtained by our coevolutionary algorithm and the best result of the genetic algorithm without grouping.

References

- AZADIVAR, F. and WANG, J., 2000, Facility layout optimization using simulation and genetic algorithms. *International Journal of Production Research*, **38**, 4369–83.
- CHAN, K. C., and TANSRI, H., 1994, A study of genetic crossover operations on the facilities layout problem. *Computers and Industrial Engineering*, **26**, 537–550.

- CHIANG, W. C., 2001, Visual facility layout design system. *International Journal of Production Research*, **39**, 1811–36.
- CONWAY, D. G. and VENKATARAMANAN, M. A., 1994, Genetic search and the dynamic facility layout problem. *Computers and Operations Research*, **21**, 955–60.
- DAS, S. K., 1993, A facility layout method for flexible manufacturing systems. *International Journal of Production Research*, **31**, 279–297.
- DE LIT, P., FALKENAUER, E. and DECHAMBRE, A., 2000, Grouping genetic algorithms: an efficient method to solve the cell formation problem. *Mathematics and Computers in Simulation*, **51**, 257–271.
- GAU, K.-Y. and MELLER, R. D., 1999, An iterative facility layout algorithm. *International Journal of Production Research*, **37**, 3739–3758.
- GERO, J. S. and KAZAKOV, V. A., 1998, Evolving design genes in space layout planning problems. *Artificial Intelligence in Engineering*, **12**, 163–176.
- HARHALAKIS, G., NAGI, R. and PROTH, J. M., 1990, An efficient heuristic in manufacturing cell formation for group technology applications. *International Journal of Production Research*, **28**, 185–98.
- KOCHHAR, J. S. and HERAGU, S. S., 1998, MULTI-HOPE: a tool for multiple floor layout problems. *International Journal of Production Research*, **36**, 3421–35.
- KOCHHAR, J. S. and HERAGU, S. S., 1999, Facility layout design in a changing environment. *International Journal of Production Research*, **37**, 2429–2446.
- MELLER, R. D. and GAU, K.-Y., 1996, The facility layout problem: recent and emerging trends and perspectives. *Journal of Manufacturing Systems*, **15**, 351–366.
- MELLER, R. D., NARAYANAN, V. and VANCE, P. H., 1999, Optimal facility layout design. *Operations Research Letters*, **23**, 117–127.
- MONTREUIL, B., 1990, A modelling framework for integrating layout design and flow network design. *Proceedings of the Material Handling Research Colloquium*, **90**(2), pp. 43–58.
- RAJASEKHARAN, M., PETERS, B. A. and YANG, T., 1998, A genetic algorithm for facility layout design in flexible manufacturing systems. *International Journal of Production Research*, **36**, 95–110.
- SCHNECKE, V. and VORNBERGER, O., 1997, Hybrid genetic algorithms for constrained placement problems. *IEEE Transactions on Evolutionary Computation*, **1**, 266–277.
- TAM, K. Y. and LI, S. L., 1991, A hierarchical approach to the facility layout problem. *International Journal of Production Research*, **29**, 165–84.
- TAVARES, J., RAMOS, C. and NEVES, J., 2000, Addressing the layout design problem through genetic algorithms and constraint logic programming. In M. H. Hamza (ed), *Artificial Intelligence and Soft Computing*. Proceedings of the IASTED International Conference, IASTED/ACTA Press, pp. 65–71.

