A COEVOLUTIONARY ALGORITHM FOR A FACILITY LAYOUT PROBLEM

T. Dunker, E. Westkämper
Fraunhofer Institute Manufacturing Engineering and Automation
Nobelstr. 12
70569 Stuttgart, Germany

G. Radons
Institute of Physics
Technical University Chemnitz
Reichenhainer Str. 70
09126 Chemnitz, Germany

ABSTRACT

This paper is dedicated to a coevolutionary approach to the numerical optimization of large facility layouts using genetic algorithms. First we introduce a new way to code the relative positions of the departments. We present improved mutation and crossover operators for this problem. In order to cope with large problem size, departments are clustered into groups. These groups evolve in separate areas while position and size of these areas undergo an evolution, too.

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 facility layout problem (FLP). The objectives shortly described by the word "good" are manifold. In addition many of them are of qualitative nature and it is not straight forward to translate them to quantitatively measurable criteria. 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. Yet, in all cases the starting assumption is that the proximity between certain departments is more favorable than others. The aim is to arrange the departments in such a way that the desired proximity is satisfied.

For several decades there has been research on this subject. [1] contains a detailed review of the different formulations of the FLP and the variety of algorithms. Some additional recent references can be found in the introduction of [2]. One can roughly distinguish between three types of problem formulations - assigning departments to a finite number of locations, subdividing the available floor area successively and placing departments of given size arbitrarily on the shop floor. This last formulation which yields a mixed integer programming problem (MIP) forms the base for our considerations.

All these approaches lead to combinatorial optimization problems and they share the difficulty that the computational complexity grows very rapidly with the number of departments. Hence many suggested algorithms for treating these problems try to find heuristically good solutions instead of aiming at global optimality of the solution. Evolutionary methods like genetic algorithms (GA) have successfully been applied in this context, see e.g. [3], [4] or [5].

In this work we want to present a new coding and genetic operators for handling the relative position of departments and a coevolutionary approach in order to attack large scale FLP's.

The remainder of this paper is organized as follows. In section 2 we describe our MIP formulation. Section 3 is dedicated to the genetic operators and the coevolutionary algorithm. Finally, we discuss the performance of our genetic operators and the numerical results of the coevolutionary algorithm in section 4.

2. MIP FORMULATION

The approximation of departments, floor areas and others by rectangles is a popular method, see e.g. [5] or [2]. In addition we assume that their sides are parallel to the axis of our coordinate system in the plane and their dimensions are given in advance. Denoting a rectangle by $A \subset \mathbb{R}^2$ we introduce the following notations:

$I = \{1, \ldots, N\}$	index set of all rectangles
$i, j \in I$	indices for the rectangles
$l_i \in \mathbb{R}_+$	length of the long side
$s_i \in \mathbb{R}_+$	length of the short side
$(X_i, Y_i) \in \mathbb{R}^2$	center point coordinates
$O_i \in \{0, 1\}$	orientation vertical/horizontal
$M_i^X \in \{0, 1\}$	flip up/down
$M_i^Y \in \{0, 1\}$	flip left/right

Let i^* be the index of the total available floor area. Later we want to place departments inside of subareas of the floor area, i.e. we require that certain rectangles are completely contained in other rectangles (which themselves could be inside of a third rectangle). In order to formalize this we introduce the following map on the index set of all rectangles $P : I \to I$, 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 containment 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 $I^{\mathsf{m}} \subset I$ and $I^{\mathsf{f}} \subset I$ with $I^{\mathsf{m}} \cap I^{\mathsf{f}} = \emptyset$ and $I^{\mathsf{m}} \cup I^{\mathsf{f}} = I$, where m stands for movable and f for fixed. Obviously, it should be $i^* \in I^{\mathsf{f}}$ and $i \in I^{\mathsf{f}}$ will mean A_i is fixed to $A_{P(i)}$.

This research was supported by DFG research project SFB 467 "Wandlungsfähige Unternehmensstrukturen für die variantenreiche Serienproduktion"

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

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

Then we can distinguish the following three cases.

1. If $i \in I^{\mathsf{m}}$ 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 $P^{\circ k}(j) = i$ for some k.
2. If $i \in I^{\mathsf{f}}$ and $P^{\circ k}(i) \in I^{\mathsf{f}}$ for all k then all quantities are constants and there is no need for M_i^X and M_i^Y.
3. If $i \in I^{\mathsf{f}}$ and $P^{\circ k}(i) \in I^{\mathsf{m}}$ for some k then set $k^* = \min\{k : P^{\circ k}(i) \in I^{\mathsf{m}}\}$ and $j = P^{\circ k^*}(i)$. 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)
$o_i^j \in \{0,1\}$	relative orientation of A_i with respect to A_j (1 corresponds to parallel long sides)

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

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

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 as the variables U_j and V_j take values in $\{0,1\}$ only. One can verify that the constraints

$$\begin{aligned} 0 &\le O_j + M_j^Y - U_j && (2)\\ 0 &\le O_j - M_j^Y + U_j && (3)\\ 0 &\le -O_j + M_j^Y + U_j && (4)\\ O_j + M_j^Y + U_j &\le 2 && (5)\\ O_j + M_j^X - V_j &\le 1 && (6)\\ O_j - M_j^X + V_j &\le 1 && (7)\\ -O_j M_j^X + V_j &\le 1 && (8)\\ 1 &\le O_j + M_j^X + V_j && (9) \end{aligned}$$

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

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

If $P(i) = j$ we can express the relation $A_i \subset A_j$ by the following four inequalities

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

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

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

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

Two non-overlapping rectangles A_i and A_j must be separated by a vertical or horizontal line. The first MIP for the FLP due to [6] proposes four binary variables for each non-overlapping relation. In [7] one optimized this approach to improve the performance. In [8] and [5] one uses a formulation which needs three binary variables per non-overlapping relation. We will propose an model using two variables which is superior by almost an order of magnitude, see table 1. We introduce

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

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 $L = \max_i l_i$. Using the four linear expressions

$$\begin{aligned} E_{ij}^{(1)} &= L\,(S_{ij}^D + S_{ij}^O) && (15)\\ E_{ij}^{(2)} &= L\,(1 + S_{ij}^D - S_{ij}^O) && (16)\\ E_{ij}^{(3)} &= L\,(1 - S_{ij}^D + S_{ij}^O) && (17)\\ E_{ij}^{(4)} &= L\,(2 - S_{ij}^D - S_{ij}^O) && (18) \end{aligned}$$

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

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

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

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

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

We denote by $I^{\mathsf{no}} \subset \{(i,j) : i > j; i,j \in I\}$ 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: $A_2 \subset A_1 = A_{i^*}$, $A_3 \subset A_1$, $A_4 \subset A_2$, $A_5 \subset A_2$, $A_6 \subset A_3$ and $A_7 \subset A_3$. In order to have the non-overlapping between A_2, A_3 and $A_4, \ldots, A_7$ it suffices $I^{\mathsf{no}} = \{(3,2),(5,4),(7,6)\}$. The relations $(6,4),(6,5),(7,4)$ and $(7,5)$ are automatically true because of the containment and $(3,2)$.

The objective is to minimize the distances between different points. We will call them IO-points. They are attached to some floor area or a department, i.e. a rectangle may possess several IO-points. We use the following notation:

branching model	automatic two	automatic three	strong two	strong three
b&b tree size ($\times 10^3$)	11.5	47.9	5	31
computation time in s	25.5	187.6	46.3	512

Table 1: Comparison of our model (two) with the model (three) from [8] or [5] (FLP with 6 departments from [8]).

$I^\bullet = \{1,\dots,N^\bullet\}$	set of indices for IO-points,
$\alpha,\beta \in I^\bullet$	indices for IO-points,
$(x_\alpha^{\bullet i}, y_\alpha^{\bullet i}) \in \mathbb{R}^2$	relative coordinates wit respect to A_i, $i \in I^{\mathsf{m}}$,
$(X_\alpha^\bullet, Y_\alpha^\bullet) \in \mathbb{R}^2$	coordinates of the α-th IO-point which are either constant (IO-point is attached to A_{i^*} or is variable satisfying e.g. equation (1) or is identical with center point of A_i),
$\delta X_{\alpha\beta}^\bullet, \delta Y_{\alpha\beta}^\bullet \in \mathbb{R}_+$	horizontal and vertical distance between the α-th and β-th IO-point,
$(w_{\alpha\beta})_{\substack{\alpha,\beta\in I^\bullet\\ \alpha>\beta}}$	weights for the distances.

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 $i \in I^{\mathsf{m}}$:

$X_i^{(0)}, Y_i^{(0)}, O_i^{(0)}$	initial position and orientation,
$M_i \in \{0,1\}$	1 if X_i, Y_i or O_i have changed from the initial layout or if M_i^X or M_i^Y equals to 1,
$(w_i)_{i\in I^{\mathsf{m}}}$	weights for moving A_i.

With this notation the objective is to minimize

$$\min\left(\sum_{\substack{\alpha,\beta\in I^\bullet\\ \alpha>\beta}} w_{\alpha\beta}(\delta X_{\alpha\beta}^\bullet + \delta Y_{\alpha\beta}^\bullet) + \sum_{i\in I^{\mathsf{m}}} w_i M_i\right) \tag{23}$$

where the following further constraints have to be satisfied

$$X_\alpha^\bullet - X_\beta^\bullet \le \delta X_{\alpha\beta}^\bullet \tag{24}$$
$$X_\beta^\bullet - X_\alpha^\bullet \le \delta X_{\alpha\beta}^\bullet \tag{25}$$
$$Y_\alpha^\bullet - Y_\beta^\bullet \le \delta Y_{\alpha\beta}^\bullet \tag{26}$$
$$Y_\beta^\bullet - Y_\alpha^\bullet \le \delta Y_{\alpha\beta}^\bullet \tag{27}$$

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

$$X_i - X_i^{(0)} \le L\,M_i \tag{28}$$
$$X_i^{(0)} - X_i \le L\,M_i \tag{29}$$
$$Y_i - Y_i^{(0)} \le L\,M_i \tag{30}$$
$$Y_i^{(0)} - Y_i \le L\,M_i \tag{31}$$
$$O_i - O_i^{(0)} \le M_i \tag{32}$$
$$O_i^{(0)} - O_i \le M_i \tag{33}$$
$$M_i^X \le M_i \tag{34}$$
$$M_i^Y \le M_i \tag{35}$$

for all $i \in I^{\mathsf{m}}$ and $w_i \neq 0$.

3. THE COEVOLUTIONARY GA

The goal to find 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 S_{ij}^D and S_{ij}^O. That is why various heuristic methods have been developed in order to find systematically at least suboptimal solutions. They try to fix some of the binary variables. One such approach are genetic algorithms, see e.g. [9], [10], [11] (quadratic assignment problem), [12], [3] (slicing tree representation), [5], [4] (MIP), [13] (slicing tree and MIP).

3.1. Coding

In [5] the 0-1-sequence for setting the binary variables is used as genetic code. Standard crossover and mutation operators are applied. This generates subsequently many infeasible genes as possible settings do not satisfy the transitivity of relative positions. In order to avoid producing too many infeasible genes we introduce a coding and operators which do not leave the space of genes satisfying transitivity.

We suppose that the problem to solve involves the rectangles with indices from the index set $I_k^{\mathsf{g}} \subset I$ where the letter g stands for “group” and $k = 1, 2, \dots$ is the index of the group. Let $N_k^{\mathsf{g}} = \# I_k^{\mathsf{g}}$ be the number of the involved rectangles. Denote by $\iota = 1,\dots,N_k^{\mathsf{p}}$ 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

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

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

Given an $\mathbb{I}_{k,\iota}^{(\gamma)}$ we set the S_{ij}^D and S_{ij}^O with $i,j \in I_k^{\mathsf{g}}$ and $i > j$ in the following way. For each pair of indices $0 \le j_1 < j_2 \le N_k^{\mathsf{g}}$ 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 then set

$$S_{i^x j^x}^D = 0 \tag{36}$$
$$S_{i^x j^x}^O = \begin{cases} 0 & \text{if } i^x = i_{j_1}^x \\ 1 & \text{if } i^x = i_{j_2}^x \end{cases}. \tag{37}$$

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 $X_{i_{j_1}^x} \le X_{i_{j_2}^x}$. Analogously, 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, then we assign

$$S_{i^y j^y}^D = 1 \tag{38}$$
$$S_{i^y j^y}^O = \begin{cases} 0 & \text{if } i^y = i_{j_1}^y \\ 1 & \text{if } i^y = i_{j_2}^y \end{cases}. \tag{39}$$

3.2. Genetic operators

In [9] three different crossover operators for permutations are presented – partially matched, order and cycle crossover. For our genetic algorithm we use a version of the order crossover. After selecting two parent genes $\mathbb{I}^{(\gamma)}_{k,\iota_1}$ and $\mathbb{I}^{(\gamma)}_{k,\iota_2}$ let us consider the parts of the genes representing the x- and y-order. Take e.g.

$$(i^x_{1,\iota_1},\dots,i^x_{N^{\mathrm{g}}_k,\iota_1}) \quad \text{and} \quad (i^x_{1,\iota_2},\dots,i^x_{N^{\mathrm{g}}_k,\iota_2})$$

where we add the number of the individual as a second subindex. We select randomly two cut positions $c_1, c_2 \in \{1,\dots,N^{\mathrm{g}}_k\}$ with $c_1 \le c_2$. 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

$$(\dots,i^x_{c_1,\iota_1},\dots,i^x_{c_{2,\iota_1}},\dots),(\dots,i^x_{c_1,\iota_2},\dots,i^x_{c_2,\iota_2},\dots).$$

Then the position before c_1 and after c_2 are filled with 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 where we take the elements from. In terms of the notation above this means e.g. for the first offspring gene

$$\begin{array}{l}(\tilde{i}^x_{1,\iota_2},\dots,\tilde{i}^x_{c_1-1,\iota_2},\\ i^x_{c_1,\iota_1},\dots,i^x_{c_2,\iota_1},\\ \tilde{i}^x_{c_1,\iota_2},\dots,\tilde{i}^x_{N^{\mathrm{g}}_k-c_2+c_1-1,\iota_2}) \quad \text{with}\end{array}$$

$$\{\tilde{i}^x_{1,\iota_2},\dots,\tilde{i}^x_{N^{\mathrm{g}}_k-c_2+c_1-1,\iota_2}\} \cap \{i^x_{c_1,\iota_1},\dots,i^x_{c_2,\iota_1}\} = \emptyset$$

and the mapping $j \mapsto f(j)$ defined by $\tilde{i}^x_{j,\iota_2} = i^x_{f(j),\iota_2}$ is strictly increasing. In the same way the crossover is defined for the y-order. 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 using the order given by the other parent.

For mutation a single parent is randomly selected. Then the mutation operator for each of the parts $(i^x_1,\dots,i^x_{N^{\mathrm{g}}_k})$ and $(i^y_1,\dots,i^y_{N^{\mathrm{g}}_k})$ just exchanges two randomly chosen elements.

The more complicate part is the modification of $\{b_{ij}\}_{i,j}$ which represents the decision whether to have a vertical or horizontal separating line. As we did not find a method which could be geometrically motivated we decided to use standard crossover with two cut positions and standard mutation. Yet, in addition we apply an improvement strategy. First, we fix the variables S^D_{ij} and S^O_{ij} for the given individual $\mathbb{I}^{(\gamma)}_{k,\iota}$ according to (36) – (39) and solve the remaining MIP (1) – (35).

```
evaluate ($\mathbb{I}^{(\gamma)}_{k,\iota}$)
    for all $i,j \in I^{\mathrm{g}}_k$ with $i > j$
        fix $S^D_{ij}$ and $S^O_{ij}$
    endfor
    solve remaining MIP
    return solution
```

Next, we update $(i^x_1,\dots,i^x_{N^{\mathrm{g}}_k})$ and $(i^y_1,\dots,i^y_{N^{\mathrm{g}}_k})$ by sorting the center point coordinates of the obtained solution. Then we check for all S^D_{ij} whether they can be changed without violating a constraint in the current solution.

```
change_is_possible ($S^D_{ij}$)
    if $S^D_{ij} == 0$ (x-direction)
        if $|Y_i - Y_j| \ge s_iO_i/2 + l_i(1-O_i)/2 +$
                          $s_jO_j/2 + l_j(1-O_j)/2$
            return true
        else
            return false
        endif
    else (y-direction)
        if $|X_i - X_j| \ge l_iO_i/2 + s_i(1-O_i)/2 +$
                          $l_jO_j/2 + s_j(1-O_j)/2$
            return true
        else
            return false
        endif
    endif
```

If possible we change the variable S^D_{ij}. These actions are repeated until the objective value does not decrease further. Summarizing we have sketched our improvement strategy below.

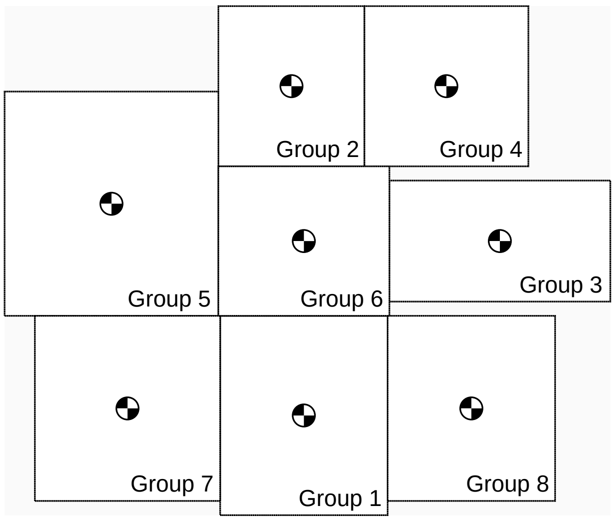
```
improve ($\mathbb{I}^{(\gamma)}_{k,\iota}$)
    while the objective value decreases
        evaluate ($\mathbb{I}^{(\gamma)}_{k,\iota}$)
        obtain $(i^x_1,\dots,i^x_{N^{\mathrm{g}}_k})$ and $(i^y_1,\dots,i^y_{N^{\mathrm{g}}_k})$
        from the solution
        for all $i,j \in I^{\mathrm{g}}_k$ with $i > j$
            if change_is_possible ($S^D_{ij}$)
                if $S^D_{ij} == 0$
                    $b_{ij} = 1$
                else
                    $b_{ij} = 0$
                endif
            endif
        endfor
    endwhile
    return $\mathbb{I}^{(\gamma)}_{k,\iota}$ with smallest objective value
```

Using these operators our GA creates a new generation by N^{cr} crossovers, N^{mu} mutations with mutation rate R^{mu} and copying the N^{co} best individuals which results in a population size of $2N^{\mathsf{cr}} + N^{\mathsf{mu}} + N^{\mathsf{co}}$. The selection accepts individuals with objective value above average with a probability of P^{ac}. The GA terminates if the average objective values has not changed more than M^{ch} or the best values has not changed for the last N^{nc} generations or a maximal number M^{ge} of generations has been exceeded.

```
genetic_algorithm
    initialize population
    do
        $\gamma = \gamma + 1$
        for $1,\dots,N^{\mathsf{cr}}$
            select parents and crossover
        endfor
        for $1,\dots,N^{\mathsf{mu}}$
            select parent and mutate
        endfor
        copy $N^{\mathsf{co}}$ best individuals
    while change of average is larger than $M^{\mathsf{ch}}$,
          best individual has changed during the
          last $N^{\mathsf{nc}}$ generations and $\gamma \le M^{\mathsf{ge}}$
```

3.3. Coevolution

For large FLP's the above described GA fails. Convergence takes very long. Hence it is necessary to treat such FLP's differently. Dividing large problems into smaller ones is a popular method, see e.g. [14]. By quantitative or qualitative method we can form groups of departments which shall be placed together. One has to provide an area for each group of departments and one has to determine the layout for each group within these areas. Already in [15] one suggests to approach the FLP 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. 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 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.

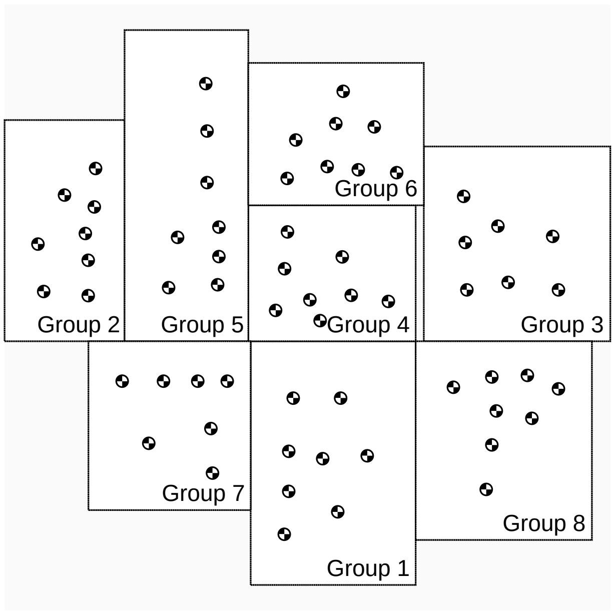


Initial group areas (246 sec.)

Figure 1: As an illustrative example we take a FLP with 62 departments clustered into 8 groups. The group areas in this initial layout are almost all square shaped.

Keeping the external IO-points for all groups constant each group passes a short evolution by a GA. 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 give a contribution, too. It is obvious that these GA's 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

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

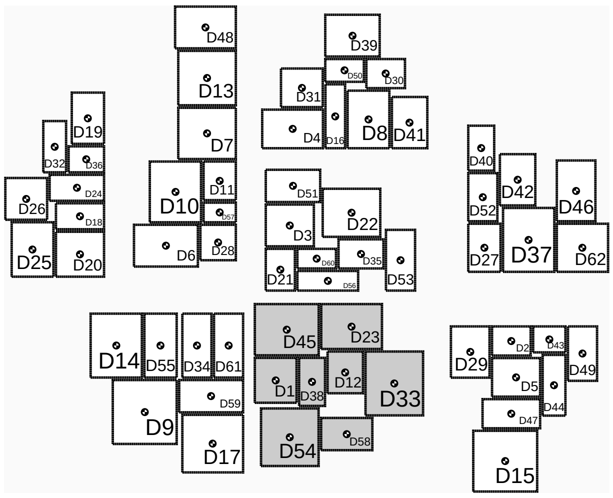


Group areas after 4 iterations (5273 sec.)

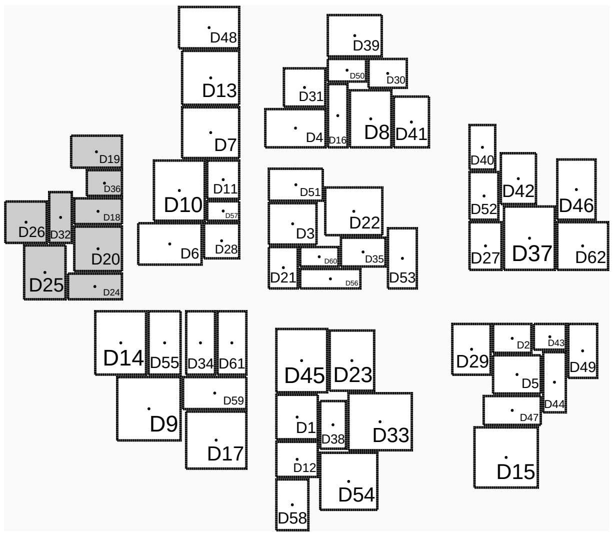
Figure 2: This layout of the group areas after 4 iterations illustrates how the shape of the group areas may change, see e.g. group 2 and 5.

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 for group $k$
        fit a group area around obtained layout
        enlarge the area by $Z_k$
    endfor
    do
        genetic_algorithm for group areas
        (treat group areas as departments with
        several IO-points)
        for all $k$ ⋆ can be done simultaneously ⋆
            genetic_algorithm for group $k$
            (placement is restricted to the group area
            and all external IO-points are fixed)
            fit a new group area around the group
            if 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
```



Iteration 5, best individual for group 1 (5343 sec.)



Iteration 5, best individual for group 2 (5408 sec.)

Figure 3: Keeping the external IO-Points (white departments) constant each group (gray departments) passes an evolution. These evolutions can be computed in parallel. The figure shows the best individuals of the fifth iteration for group 1 and 2.

4. RESULTS AND CONCLUSIONS

First let us discuss the quality of our coding and the genetic operators. Applying them to the FLP with 8 departments from [8] we obtained satisfactory results. Running the deterministic algorithm (31 h 30 min on a Pentium III 866 MHz and memory use ca. 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. In all cases computations took less than 10 minutes (on a Pentium II 400 MHz. Also for the FLP with 10 and 12 departments our approach shows very good results. Table 2 compares our best results to the best known from [8] and [5].

N	Four-Step	GA 1998	GA 2002
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 FLP's from [8] which are reported in [8] (Four-Step) and [5] (GA 1998) in comparison with our best results (GA 2002).

For testing the coevolutionary GA we created a random example with 62 departments (FLP62) of different shapes. For simplicity we placed one IO-point in the center of each department. The weights $w_{\alpha\beta}$ in (23) were generated randomly, too. There is no initial layout. Hence, all the weights w_i are zero. A table with all quantities can be obtained from the authors[1].

In order to find a grouping we implemented the heuristic grouping algorithm from [16] which uses only simple exchange operations. A similar clustering algorithm was applied in [15]. A GA proposed in [17] can generate improved groupings. The aim is to arrange the departments into groups minimizing the sum of the weights between departments belonging to different groups. In addition, one restricts the size of each group to a limit. We have tested an example with 6 groups with a maximum of 11 departments each and, secondly, 8 groups with at most 8 departments.

For the parameters introduced in section 3.2 and 3.3 we used the following settings: $R^{\text{mu}} = 10\%$, $P^{\text{ac}} = 20\%$, $N^{\text{nc}} = 5$, $M^{\text{ch}} = 0.5\%$

	N^{cr}	N^{mu}	N^{co}	M^{ge}
first area layout	20	5	5	15
area layout	0	5	5	1
group layouts	12	3	3	3

and initial enlargement factor $Z_k = 60\%$, 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.

Running the two examples with our coevolutionary algorithm on a PC (Pentium II 400 MHz) 10 times we obtained the following results:

	6 groups	8 groups
best objective	$4\,167\,956.8$	$4\,296\,388.2$
worst objective	$4\,387\,725.0$	$4\,550\,281.4$
average	$4\,304\,970.7$	$4\,411\,970.2$
average time	5 h 30 min	6 h 23 min
worst time	7h 34 min	10 h 57 min

It turned out that the computations treating the part where the 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 six group setting performed in general faster than the eight group one.

[1]email: tmd@ipa.fhg.de

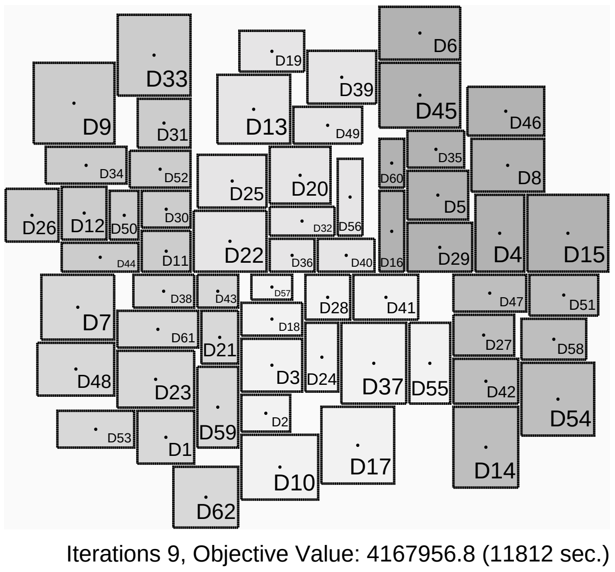


Figure 4: Best solution for FLP62 clustered into 6 groups obtained by our coevolutionary GA.

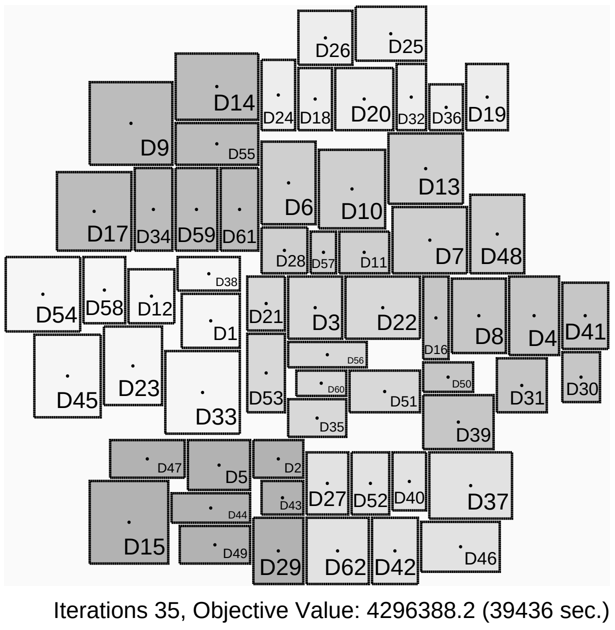


Figure 5: Best solution for FLP62 clustered into 8 groups obtained by our coevolutionary GA.

We hoped to get some reference solutions by saving incumbent solutions of a branch and bound solving FLP62. Yet, even after one week of computation on a PC with a Pentium III 866 MHz there was no result. A simple GA coped much better with the size of the problem. It found a solution (4 221 911.0) in the same range as the coevolutionary GA. Yet, it took more than one week of computation.

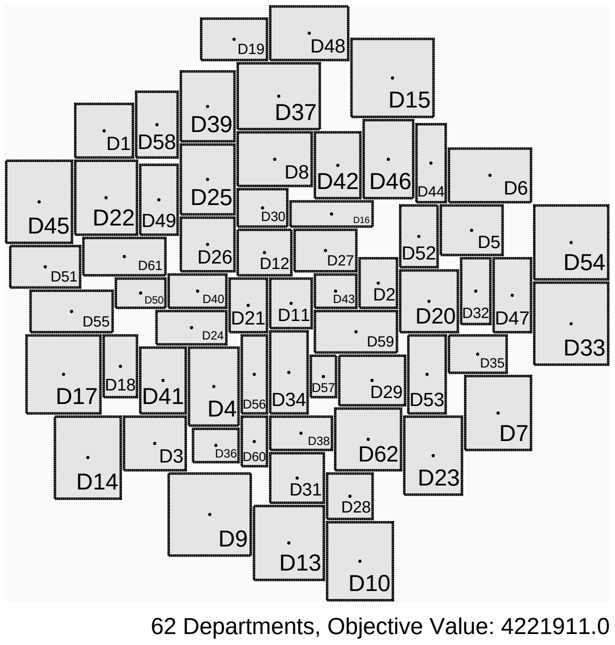


Figure 6: Solution for FLP62 from a single run of a simple GA which took 9 days an 18 hours on a PC with a Pentium II 400 MHz.

Summarizing, we can conclude that the proposed coevolutionary GA opens up the possibility to find good solutions for large FLP's within some hours where global optimization algorithms fail. In addition there is still a high potential for further computational acceleration by parallelization.

5. REFERENCES

[1] R. D. Meller and K.-Y. Gau. The facility layout problem: Recent and emerging trends and perspectives. *Journal of Manufacturing Systems*, 15(5):351–366, 1996.

[2] W. C. Chiang. Visual facility layout design system. *International Journal of Production Research*, 39(9):1811–36, 2001.

[3] F. Azadivar and J. Wang. Facility layout optimization using simulation and genetic algorithms. *International Journal of Production Research*, 38(17):4369–83, 2000.

[4] J. Tavares, C. Ramos, and J. Neves. Addressing the layout design problem through genetic algorithms and constraint logic programming. In M. H. Hamza, editor, *Artificial Intelligence and Soft Computing. Proceedings of the IASTED International Conference*, pages 65–71. IASTED/ACTA Press, 2000.

[5] M. Rajasekharan, B. A. Peters, and T. Yang. A genetic algorithm for facility layout design in flexible manufacturing systems. *International Journal of Production Research*, 36(1):95–110, 1998.

[6] B. Montreuil. A modelling framework for integrating layout design and flow network design. In *Progress in material handling and logistic / Material Handling '90*, pages 95–116, 1991.

[7] R. D. Meller, V. Narayanan, and P. H. Vance. Optimal facility layout design. *Operations Research Letters*, 23(3-5):117–127, 1999.

[8] S. K. Das. A facility layout method for flexible manufacturing systems. *International Journal of Production Research*, 31(2):279–297, 1993.

[9] K. C. Chan and H. Tansri. A study of genetic crossover operations on the facilities layout problem. *Computers and Industrial Engineering*, 26(3):537–550, 1994.

[10] J. S. Gero and V. A. Kazakov. Evolving design genes in space layout planning problems. *Artificial Intelligence in Engineering*, 12(3):163–176, 1998.

[11] J. S. Kochhar and S. S. Heragu. Multi-hope: a tool for multiple floor layout problems. *International Journal of Production Research*, 36(12):3421–35, 1998.

[12] V. Schnecke and O. Vornberger. Hybrid genetic algorithms for constrained placement problems. *IEEE Trans. Evolutionary Computation*, 1(4):266–277, 1997.

[13] K-Y. Gau and R. D. Meller. An iterative facility layout algorithm. *International Journal of Production Research*, 37(16):3739–3758, 1999.

[14] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi. Scaling up fast evolutionary programming with cooperative coevolution. In *Proceedings of the 2001 Congress on Evolutionary Computation*, volume 2, pages 1101–8, 2001.

[15] K. Y. Tam and S. L. Li. A hierarchical approach to the facility layout problem. *International Journal of Production Research*, 29(1):165–84, 1991.

[16] G. Harhalakis, R. Nagi, and J. M. Proth. An efficient heuristic in manufacturing cell formation for group technology applications. *International Journal of Production Research*, 28(1):185–98, 1990.

[17] P. De Lit, E. Falkenauer, and A. Dechambre. Grouping genetic algorithms: an efficient method to solve the cell formation problem. *Mathematics and Computers in Simulation*, 51(3-4):257–271, 2000.