Incorporating Fuzzy Knowledge into Fitness: Multiobjective Evolutionary 3D Design of Process Plants

Ingo Mierswa
Artificial Intelligence Unit
Department of Computer Science
University of Dortmund
ingo.mierswa@uni-dortmund.de

ABSTRACT

Designing technical plants is a complex and demanding process. It has been shown that the optimization of the simple facility placement problem is already NP-hard. Optimization of plant designs must obey a number of criteria derived from several fields of process engineering. We discuss an expansion of the simple facility placement problem with non-regular floor spaces and more than one layer. Additionally, we allow forbidden zones and predefined ways. In contrast to other approaches our system can cope with competitive criteria. These can be defined by a plant designer in an intuitive way according to concepts from fuzzy logic. This leads to the multiobjective optimization of costs and fulfillment of weighted design rules. We describe an evolutionary algorithm to construct Pareto-optimal blueprints of chemical plants. The smart indexing of rules and assignment of conclusions to components allows an efficient calculation of the rule fulfillment as part of the fitness function. Optimized blueprints for a real existing chemical plant dominate the original design.

Categories and Subject Descriptors: I.2 [Computing Methodologies]: Artificial Intelligence

General Terms: Experimentation

Keywords: Multiobjective evolutionary construction, evolutionary plant design, fuzzy constraint indexing

1. INTRODUCTION

Designing a process plant is a complex and demanding task. Right from the planning stage the designer must consider the complete lifecycle of the plant and costs for realization and operation. Wrong decisions in these stages cause high costs during installation and operation. First of all, the designer selects necessary components for the process at hand. The optimization of this selection is done during *basic engineering* and is not part of this paper. After this selection step the main components and their properties are known and must be placed on a factory ground. This design phase is called *extended basic engineering* and all following steps of *detail engineering* depend on an optimal placement of the selected components. Factory grounds often contain a steel construction which expands the layout problem into the third dimension. In this paper we concentrate on the design of chemical plants where a steel construction usually exist. Typical dimensions of the used equipments cover the range from $1m^2$ to $50m^2$. The layout optimization in the extended basic engineering step must obey a great number of constraints derived from several fields of engineering [2, 21]. *Process engineering* may define requirements depending on the process at hand, e. g. a pump may ask for a primary pressure which can be set with help of an altitude difference. *Safety engineering* defines constraints which should ensure a safe operation for both the plant and the employees. Placement constraints for components can be derived from *machine engineering*, e. g. heavy machines should be placed in ground level. All relevant *pipes* should be considered during planning, especially those with large diameters or those built from expensive materials. The *installation requirements* should ensure a low priced installation of the plant. Components which must be manually *operated* or *maintained* should be placed at reachable locations. These requirements can be defined as rules (see section 3). A blueprint for a process plant is optimal if all rules are fulfilled and other costs are minimized. Often these constraints can only be formulated in a fuzzy way and, moreover, are usually contradictory. For example, a plant providing enough place for maintaining cannot minimize the connection lengths. In practice an optimal plant design can hardly be found because of the competitive criteria and costs defined above.

1.1 Related work

The optimization of process plants discussed in this paper is a variant and enhancement of known problems. The placement problem is known from VLSI design, where the total space needed by a set of rectangular components is to be minimized. This problem is NP-hard [8]. Heuristics must be used to ensure efficiency in practical domains, including evolutionary algorithms [6, 10, 11, 29].

Similar to the placement problem of VLSI design is the facility placement problem. Classical methods for this problem include the definition as quadratic assignment problem, linear programming, and mixed integer linear programming [19]. The main problem with these approaches is the reduction on a single class of costs [1]. Improvements were done to overcome this problem [22, 25] but finding an exact solution with respect to general cost functions remains a hard problem [26]. Other approaches relax this problem by using a fixed grid [14]. Since the plant components used in this paper are very different in size this restriction is not an option. None of the approaches optimize the layout with respect to all criteria defined above.

Many solutions based on evolutionary algorithms were presented including approaches which uses several weighted criteria [4] for required area, pipe lengths, and safety costs. The proposed approaches did not perform a multiobjective optimization and do not allow three dimensions or layers. Additionally, the user can not define additional constraints which may depend on the process at hand. Other evolutionary approaches did not allow fuzzy user defined constraints too [30, 3, 31]. Hence we still lack an efficient knowledge-based assistant for three dimensional real world plant design.

1.2 Outline

In the next section we formalize the layout problem discussed in this paper as enhancement of the classical facility layout problem. We discuss a representation of plants which allows the definition of fuzzy constraints and can be used to represent the individuals during optimization. Afterwards, we will introduce the indexing of fuzzy rules in section 3. This is a necessary preprocessing step to guarantee an efficient calculation of the fitness function. Users can easily define the constraints of plant design in a simple XML format. In section 3.1 we will discuss the possible rule conclusions which are defined with fuzzy techniques. In Section 4 we merge the different criteria like connection costs (section 2.1) and rule fulfillment (section 3.1). We will also shortly introduce some other criteria which were not discussed in detail. This leads to the multiobjective optimization problem of 3D plant design using fuzzy background knowledge. We use an evolutionary strategies approach described in section 4.1 on the individuals defined in section 2 to tackle this problem. We adapted Pareto dominant tournament selection for the multiobjective setting where no criteria weights were given. Section 5 provide results for an existing plant for both the single- and multiobjective setting.

2. ENLARGEMENT OF THE CLASSICAL FACILITY LAYOUT PROBLEM

In the classical facility layout problem we try to arrange a set of rectangular shapes in a specified area [28]. At the same time the costs induced by the matrix $M = [m_{ij}]_{i,j=1,\ldots,n}$ should be minimized. This matrix defines the costs for a connection between the components i and j. The matrix entry m_{ij} depends on the diameter and the material of the connection. The goal is a non-overlapping layout of all components with minimized costs

$$\sum_{i,j} d_{ij} \cdot m_{ij} \tag{1}$$

with d_{ij} as distance between the components i and j (see section 2.1). The facility layout problem is a special case of the quadratic assignment problem, which proved to be NP-hard [13]. We discuss a variant of this problem with non-regular floors, three dimensions by using a steel construction for higher levels, forbidden zones, and predefined ways. Similar variants were studied by [19, 23].

We represent all parts of the plant which may be relevant to user defined constraints as *components*. We do not distinguish between a way and a machine for example: both are represented as component of the plant. This allows the described and further enhancements as well as user defined constraints. All components which are movable during the design phase build a vector which is the representation of the points of our search space. Components are defined by a shape and a location. In our experiments we use only rectangular shapes, but please note that our system is not limited to these shapes. The location is defined by a triple $(x,y,layer)$. Ways and blocked zones are special components which can not be moved but are necessary to define the rules and constraints in a straightforward manner. Unless otherwise noted we refer to the movable machines of a plant if we speak of components. This component-based concept of process plants including user defined optional components covers the complex optimization task discussed in this paper as well as simple facility layout problems.

2.1 Routing approximation by Manhattan distance

We have to determine the costs for connections by building the sum $\sum_{i,j} d_{ij} \cdot m_{ij}$ using a cost matrix M. Therefore, we have to efficiently calculate the distances d_{ij} between the components i and j. This routing problem is usually solved with variants of the algorithm introduced by Lee [20]. However, the 3D routing is already NP-hard for only two layers [33]. Since the costs for connections must be calculated for each fitness evaluation we need an approximation for the distance which can be efficiently calculated. Fortunately, it turns out that the Manhattan distance is a very good approximation for the pipes of a chemical plant. The Manhattan distance is the shortest orthogonal connection of two points in space. Due to the triangle inequality in euclidian spaces the minimization of the Manhattan distance is equivalent to the minimization of the euclidian distance. Since pipes of chemical plants are almost exclusively orthogonally layed the Manhattan distance is actually a better estimation for the real costs than the euclidian distance and the calculations needs less runtime than more complicate routing algorithms. Furthermore, we do not have to consider routing collisions.

3. RULES FOR PLANT DESIGN

We motivated and discussed some of the requirements for plant design already in section 1. This section describes how the different constraints can be formalized and evaluated. We used a rulebase developed by Leuders [21] who defined about 100 rules for chemical plant design. Of course similar rulebases can be defined for other purposes too. The rulebase can be divided in two groups:

absolute position constraints: absolute constraints refer to a movable component and one of the general properties of the plant like ways or forbidden zones. An air cooler, for example, must be placed in the top level of the plant, other components must be placed next to ways to ease maintenance.

relative position constraints: relative constraints correlate two movable components without defining the absolute positions of the components. For example, a tank with a connection to a pump should be placed above the pump.

It is possible to specify both groups with the same function class due to the component based representation for both movable machines and other plant elements like ways or steel constructions. Absolute position constraints like `next to way` or `next to border` can also be defined as relative constraints between a normal component and a non-movable component like another plant element. Hence, we can define the same conclusions for both groups. These conclusions are `next`, `close`, `above`, `in`, and `out`. The fulfillment of each of these conclusions can be calculated like the membership function of a linguistic variable. In the next section this fuzzy technique is defined for the described conclusions.

3.1 Fulfillment of rules with fuzzy conclusions

There are two good reasons for using fuzzy constraints as part of the fitness function. The background knowledge which is used to define the rulebase for the fitness function should reflect the experience of a plant designer. The well known fuzzy logic concept of a linguistic variable is a natural representation of such knowledge [9]. The main reason, however, is the smooth form of the fitness function which is induced by the membership function of fuzzy constraints. In contrast to sharp logic representations the fuzzy constraints can easily be weighted and avoid discontinuous fitness functions which are more problematic to optimize at all.

Following the concept of a linguistic variable we define functions which map two components on the membership to a concept like “neighborhood” or “vertical alignment”. This is called a fuzzy constraint:

Definition 1. Let K be the set of all components. A *fuzzy constraint* is a mapping $f : K \times K \rightarrow [0,1]$ of two components on a membership interval.

Fuzzy constraints can be used as conclusion of user defined rules. Another advantage is the intuitive weighting by scaling the membership value with the rule weight. The fulfillment e_r of a rule's conclusion for two components k_i and k_j is the membership value of the corresponding fuzzy constraint c_r multiplied with the rule's weight $w_r \in \mathbb{R}$, i. e.:

$$e_r\left(k_i, k_j\right) = w_r \cdot c_r\left(k_i, k_j\right) \tag{2}$$

In many cases a fuzzy constraint does not only apply for exactly two components but for one component and a class of other components. An example is a conclusion like “the component must be next to a way”. This constraint is fulfilled if the component at hand lies next to an arbitrary way. Please remember that each way is also defined as component. We can use a class *Ways* of components and claim that the fuzzy constraint $next(k, Ways)$ should provide a high membership value if the component k lies next to one of the ways out of *Ways*. We formalize this idea of disjunction with *fuzzy class constraints*:

Definition 2. Let K be the set of all components and $C \subset K^p$ a class of components. A *fuzzy class constraint* is a mapping $\hat{f} : K \times C \rightarrow [0,1]$ of a component and a component class on a membership interval. The function $\hat{f}$ is defined as $\hat{f}(k,c) = \max_i(f(k,c_i))$ for a $k \in K$ and all $c_i \in C$.

Please note that other *S-norms* can also be used instead of the maximum.

3.1.1 Fuzzy conclusions for plant design

Now we can use the idea of fuzzy constraints to define possible conclusions of weighted design rules. These should reflect the requirements induced by process and safety engineering. Especially for chemical processes a vertical alignment of two components is often demanded. Since the exact vertical arrangement is actually not necessary we define a fuzzy constraint named `above` depending on the angle between the components. Let d_{ij} be the distance and h_{ij} the altitude difference between the components i and j. The value h_{ij} is negative if component i lies in a lower level than component j. The following fuzzy constraint delivers 0 for all angles greater than α_{max}:

Definition 3. The fuzzy constraint `above` is defined as:

$$above(i,j) = \begin{cases} 0 & \text{for } h_{ij} < 0 \\ \max\left(0, 1 - \frac{\arctan \frac{d_{ij}}{h_{ij}}}{\alpha_{max}}\right) & \text{for } h_{ij} \geq 0 \end{cases} \tag{3}$$

If two components should lie side by side in the same layer the distance d_{ij} can be normalized by the diameter d_2 of the corresponding layer:

Definition 4. The fuzzy constraint `next` is defined as:

$$next(i,j) = \begin{cases} 0 & \text{for } h_{ij} \neq 0 \\ 1 - \frac{d_{ij}}{d_2} & \text{for } h_{ij} = 0 \end{cases} \tag{4}$$

Similar to `next` which only makes sense in 3D settings with discrete layers the fuzzy constraint `close` defines the neighborhood in all dimensions with help of the diagonal d_3 of the used space:

Definition 5. The fuzzy constraint `close` is defined as:

$$close(i,j) = 1 - \frac{d_{ij}}{d_3} \tag{5}$$

Additional special purpose constraints can easily be added in a similar way. In order to demand the placement of a specific component as part of the steel construction or a particular layer the following fuzzy constraints were defined for chemical plant design:

Definition 6. The fuzzy constraints `in` and `out` are defined as:

$$in(i,j) = o(i,j)/\min(a(i), a(j)) \tag{6}$$

$$out(i,j) = 1 - in(i,j) \tag{7}$$

The function $a(i)$ calculates the area of the shape of component i and the function $o(i,j)$ the size of the intersection between the components i and j. The value range of this overlap function lies between 0 and $\min\left(a(i), a(j)\right)$.

Sometimes the user is able to define weights for the different rules. In case of the chemical plant design task we discuss in this paper the designer was able to define three different degrees of importance. Some rules must be obeyed to ensure the correct functioning of the plant or the safety of the employees. For other constraints it would only be nice if they are fulfilled but this is not really necessary. These degrees of rule importance can be defined via rule weights. Figures 1 and 2 show parts of the rulebase for chemical plant design. The used weights are named *high*, *medium*, and *low*. The complete rulebases are stored in an XML format.

```
// Long tanks should be placed next to a border
<rule name="L71" weight="low" constraint="Border">
  <method name="type" type="=" value="tank"/>
  <method name="length" type=">" value="500"/>
</rule>
// Stations should be placed in level 0
<rule name="L80" weight="low" constraint="Ground">
  <method name="type" type="=" value="station"/>
</rule>
// A short forklift must be placed next to a way
<rule name="L05" weight="high" constraint="Way">
  <method name="device" type="=" value="forklift"/>
  <method name="length" type="<" value="5000"/>
</rule>
```

Figure 1: Part of the XML rulebase for absolute position constraints. "Border" is a shortcut for the class of all border components, i. e. for the constraint next$(k, Border)$.

3.1.2 *Rulebase preprocessing and indexing*

Such XML rulebases can be specified for a complete field of technical engineering, e.g. chemical plant design. They apply for chemical plants in general and only little adoptions should be necessary when a concrete plant should be designed. On the one hand this is a desired feature since a system providing aid to a designer should have only one setup phase. On the other hand this means that many of the rules might not be applicable to a given plant at all. The naive approach for fitness evaluation is to iterate through all rules and check if the current rule applies for one or several of the components and their connections. Since this approach is far away from an efficient implementation it cannot be used during the fitness evaluation of an evolutionary algorithm. Therefore, we apply a preprocessing step on the rulebase before we start with optimization. We perform the following steps:

- Iterate through all rules and do:
  1. Iterate through all components and connections and check if the premises of the current rule can be applied
  2. Assign the conclusion, i. e. the fuzzy constraint of a firing rule to the corresponding components and connections respectively

This simple rule indexing step reduces the number of constraints to those which are suitable to the given plant at all. Moreover, the premises are checked only once before optimization and not for each fitness evaluation. We illustrate this indexing step with a simple example. The generic rulebase for chemical plant design contains a rule

```
If
a connection exists from a heat exchanger to a pump
then
the heat exchanger must be located above the pump.
```

The rule need not to be checked at all during optimization if the plant at hand does not contain any heat exchangers or pumps. If there are heat exchangers and pumps which are connected in the specified direction then only the rule conclusion must be evaluated for exactly these components.

```
// Clear tanks should be placed above pumps
<rule name="L40" weight="medium" constraint="Above">
  <first>
    <method name="type" type="=" value="tank"/>
    <method name="specs" type="=" value="clear"/>
  </first>
  <second>
    <method name="type" type="=" value="pump"/>
  </second>
</rule>
// Components must be placed close if connected
// with metal connections of large diameter
<rule name="L02" weight="high" constraint="Close">
  <connection>
    <method name="diameter" type=">" value="5"/>
    <method name="material" type="=" value="met"/>
  </connection>
</rule>
```

Figure 2: Part of the XML rulebase for relative position constraints. Arbitrary combinations of premises for both the components and the connection itself can be defined.

Rule indexing derives a concrete rule from the generic rulebase for a singular heat exchanger `e` and a singular pump `p`:

```
The heat exchanger e must be located above pump p.
```

Hence this rule need not to be checked for components of other types too. By indexing the rulebase and assigning the conclusions to the corresponding components the effort for rule evaluation is minimized.

3.1.3 *Runtime analysis of rule evaluation*

Let n be the number of components. If each component is connected to each other component a total number of $O(n^2)$ connections exist. Since premise checks and indexing is done during a preprocessing step the calculation of the membership function is the only thing we have to do for each connection. This can be done in $O(1)$. Only a constant number of possible constraints exist for each connection – in case of chemical plant design the five constraints `next, close, above, in, out`. This results in a worst case runtime of $O(kn^2)$. Practically, not all components are connected to each other but each component provide only a small amount of connections. In case of chemical plants this number is approximately 4.

4. PLANT DESIGN AS MULTIOBJECTIVE OPTIMIZATION

The rules we described in the last section cover the operational requirements discussed in section 1. However, more than these criteria may be applied to the optimization task of plant design. All of them must be efficient to calculate, i. e. have polynomial runtime. We define the following criteria for the design optimization of chemical process plants:

f_a: the fulfillment of the absolute position constraints. The runtime is linear in the number of components.

f_r: the fulfillment of the relative position constraints. The runtime is linear in the number of connections (see section 3.1.3).

f_l: the distribution of layer contents. Approximately between 30% and 70% of each layer should be filled to ease maintenance and operation. Actual values depend on the type of the plant [18]. Since each component must be examined the runtime is also linear.

f_c: connection costs based on Manhattan distance and cost matrix M (see section 2.1). The quadratic runtime in the worst case is not reached in practice. In average the complexity is similar to the calculation of relative rule fulfillment.

f_o: the overlap degree. Overlapping components are a relaxation of the classical facility placement problem. This eases a continuous mutational drift of components into other areas of the plant – including the drift across forbidden zones or ways. Since the overlap of each component with all other components must be calculated the runtime is quadratic in the number of components.

It can easily be shown that these criteria compete. A plant that fulfills all operational requirements can usually not minimize the connection costs. On the other side a strongly overlapping plant would minimize those costs. An experienced plant designer may be able to define weights w_i for the different criteria and transform the optimization task into one with a single criterion

$$Z = w_a f_a + w_r f_r + w_l f_l + w_c f_c + w_o f_o \tag{8}$$

which is to maximize. Usually such weights cannot be defined and vary with different plants. Therefore, the task of 3D plant design is a *multiobjective optimization problem*. These problems cannot be solved with singleobjective evolutionary algorithms. Since the user has only a vague idea of criteria weights, a multiobjective optimization algorithm tries to find all solutions which are optimal for arbitrary weight vectors. These solutions are called *Pareto-optimal* (see section 4.1.1). The blueprints of the Pareto-optimal set of solutions can only be enhanced with respect to one criterion if the values of other criteria decreases. We describe an evolutionary approach to find such Pareto-optimal blueprints of process plants in the next section.

4.1 Evolutionary search for Pareto-blueprints

This section describes the developed operations to perform the search for optimal plant designs with an evolutionary algorithm [15, 16]. Each individual is given as vector of the positions and orientations of all movable components (section 2). Since the shape of a component cannot be changed in the setting of plant design, search operations only refer to the location and the rotation of the components. We adapted the NSGA-II algorithm for these type of real-valued individuals [7].

The *start population* of N individuals is randomly created. We perform three steps for each component of an individual: first, we randomly select the layer of the component by selecting one of the possible layers defined by a steel construction. Second, we chose a position in this layer. The last step is to select one of the four possible rotations. All selections were done uniformly distributed. This leads to plant designs where components may overlap.

We apply *uniform crossover* by changing the position and the rotation of the parent's components. This is done for each component with probability 0.5. If an individual has n movable components one of the following *mutations* is performed with probability $1/n$ for each component:

Drift: the component drifts in a randomly chosen direction. The length of the drift vector is a Gaussian variable with standard deviation σ. This parameter is adapted during evolution by an 1/5-rule [27].

Layer: the component's location is randomly changed to a location in another layer.

Rotation: the component is rotated by $90°$, $180°$, or $270°$. Since our algorithm considers the connection points of the components, rotations may decrease connections costs.

We apply the parts of the fitness function described in the first sections of this paper on the individuals. If a weight vector of the criteria was defined we transform the fitness function into the single objective function and optimize equation 8. In this case we tried both *roulette wheel selection* and *tournament selection*. Tournament selection proved to work much better in our setting. We stop the algorithm after a user defined number of generations or after a given number of generations without improvement.

To allow non-singular fitness functions in the multiobjective case we must modify the selection operator. This is described in the next section.

4.1.1 Pareto-dominant selection

The multiobjective search space of a maximization problem is subject to a partial order:

Definition 7. A solution a *dominates* a solution b (written as $a \succ b$) if for the p criteria r_i the following is true:

$$\begin{aligned} \forall i \in \{1,\ldots,p\}: &\quad r_i(a) \geq r_i(b) \quad \wedge \\ \exists i \in \{1,\ldots,p\}: &\quad r_i(a) > r_i(b). \end{aligned} \tag{9}$$

Our selection scheme needs to decide if a solution is dominated by a set B of solutions. We define:

Definition 8. A solution a is *non-dominated* by a set of solutions B if the following is true:

$$\nexists b \in B: \quad b \succ a. \tag{10}$$

Now we are able to define what we mean with Pareto-optimal solutions:

Definition 9. A solution a is *Pareto-optimal* if a is non-dominated by the complete solution space.

Multiobjective evolutionary algorithms can optimize more than one target function by introducing special selection operators [5, 34]. Traditional approaches in the field of mathematical programming must be applied more than once for multiobjective optimization [12, 32]. Due to the population based approach of evolutionary algorithms a broad selection of Pareto-optimal solutions can be found during one run. The user can select one of these solutions after optimization. Additionally, multiobjective evolutionary algorithms do not strongly depend on form and continuity of the Pareto-optimal set [5]. The used NSGA-II approach employ a selection technique which first sorts all individuals into levels of non-domination. Until the desired population size is reached, individuals from the first levels are added to the next generation. Before adding individuals from the last possible level, this level is sorted with respect to the crowded distance to preserve diversity in the population.

Requirement	original	$(1+1)$	$(\mu+\lambda)$	multi
absolute	0.75 (6)	0.76 ± 0.01 **(2)**	0.76 ± 0.02 (3)	**0.78** (3) ± 0.02
relative	0.47 (0)	0.53 ± 0.01 (1)	**0.54** ± 0.01 **(0)**	0.53 (0) ± 0.01
layer	0.50	0.47 ± 0.09	**0.51** ± 0.09	0.51 ± 0.04
tubes	0.78	0.80 ± 0.02	0.82 ± 0.01	**0.83** ± 0.01
overlap	0.58	0.80 ± 0.01	**0.83** ± 0.01	0.79 ± 0.01
total	0.56	0.63 ± 0.01	**0.63** ± 0.01	0.63 ± 0.01

Table 2: Average results for the different optimization strategies from 20 runs. All criteria were normalized between 0 and 1 and should be maximized. The best results of each line were printed in a bold style. The values in parentheses are the number of highly weighted constraints which were not fulfilled in the best of the 20 runs.

Property	**Value**
number of components	28
predefined locations	5
movable components	23
number of connections	108
layers	5
steel construction	yes
forbidden zones	yes
ways	yes
total number of rules	88 in rulebase **(11968)**
applicable rules	**141**

Table 1: The properties of the original plant. The rulebase had a size of 88 rules which sums up to 88·(28+108)=11968 rule checks. Actually, only 141 constraints were checked during fitness evaluation.

5. EXPERIMENTS AND RESULTS

The experiments discussed in this paper were performed on a chemical plant which was already built. This allows the comparison with a existing plant instead of synthetical data. Table 1 collects the properties of this plant. Without rule indexing the 88 design rules must be checked for each component (absolute position constraint) and connection (relative position constraint). All in all a number of $(28 + 108) \cdot 88 = 11968$ design rules exists. We were able to reduce the number of fuzzy constraints which have to be checked to 141 (approximately 1.2%) by the preprocessing step and rule indexing. The evolutionary optimization of the plant design was performed with three strategies:

1. (1+1)EA with population size 1 and without crossover.

2. $(\mu + \lambda)$EA with population size 12 and with uniform crossover and tournament selection with $t = 4$.

3. multiobjective optimization with the selection scheme described in section 4.1.1 and population size 300.

All runs were performed with 10000 generations. Table 2 shows the average fulfillment of the constraints and the average values of the other criteria for the original plant and for the different strategies, collected during 20 runs. We have used the weights $w_a = 0.24$, $w_r = 0.60$, $w_l = 0.02$, $w_c = 0.02$, and $w_o = 0.12$ for the calculation of the weighted average in the case of the single objective optimization. The results presented for the multiobjective optimization are from the plant design which optimize the total costs for these weights. Figures 3 and 4 show the best individual optimized with the $(\mu + \lambda)$EA. During the multiobjective run a set of very diverse blueprints is created which is a useful aid for the plant designer. Figure 5 presents the initial populations and the Pareto fronts for four of the five criteria. Further results are presented in [24] and a comparison with a classical constraint programming approach can be found in [17]. This approach finds designs with similar performance with respect to the rule constraints but does not consider the other criteria.

6. CONCLUSIONS

We have introduced a way of incorporating fuzzy background knowledge into the fitness function of an evolutionary algorithm. A preprocessing step and indexing the rulebase allow the efficient calculation of the fulfillment as part of the fitness function. Special search operators were defined for the representation of process plants and a multiobjective selection scheme was adapted. Thereby we were able to cope with a variant of the facility placement problem which was proven to be NP-hard. This variant allows non-regular grounds, three dimensions, forbidden zones, and predefined ways and component locations. Our approach constructs blueprints which beats a real existing plant with respect to all criteria. Therefore this is a valuable example of a real world application of multiobjective evolutionary optimization.

7. ACKNOWLEDGMENTS

This work was supported by the *Deutsche Forschungsgemeinschaft (DFG)* within the *Collaborative Research Center "Design and Management of Complex Technical Processes and Systems by Means of Computational Intelligence Methods"*.

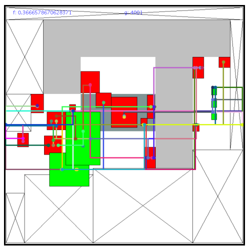
8. REFERENCES

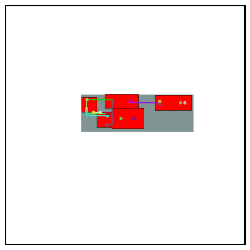
[1] L. Amorese, V. Cena, and C. Mustacchi. A heuristic for the compact location of process components. *Chemical Engineering Science*, 2(32):119–124, 1977.

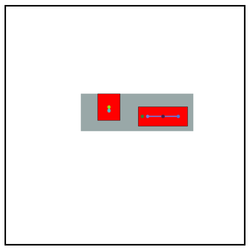
[2] W. D. Baasel. *Preliminary Chemical Engineering Plant Design*. Elsevier, 1974.

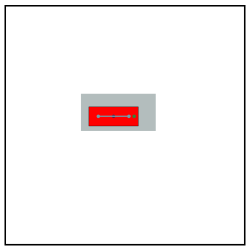
[3] J. Balakrishnan, C. H. Cheng, D. G. Conway, and C. M. Lau. A hybrid genetic algorithm for the dynamic plant layout problem. *International Journal of Production Economics*, 86:107–120, 2003.

[4] C. M. L. Castell, R. Lakshmanan, J. M. Skilling, and R. Bañares-Alcántara. Optimisation of process plant


(a) Layer 0


(b) Layer 1


(c) Layer 2


(d) Layer 3

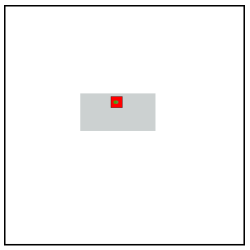
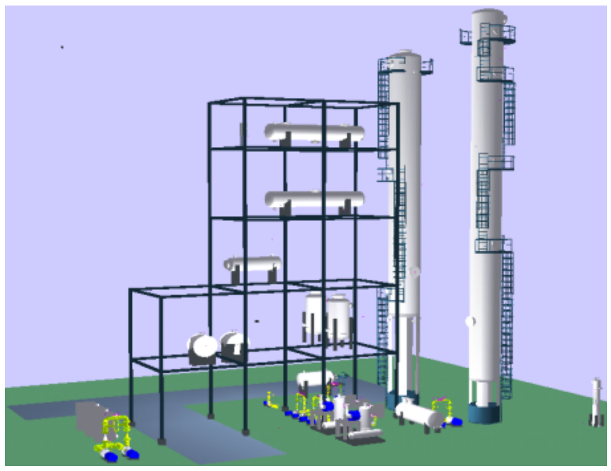

(e) Layer 4

Figure 3: The layers of the optimized plant design. Gray regions represent the ways, the predefined steel construction, and movable and predefined components. Dots of components mark the connection points.


(a) Original plant design

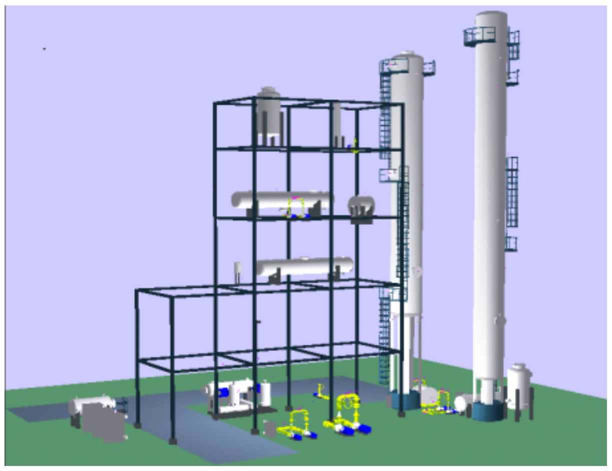
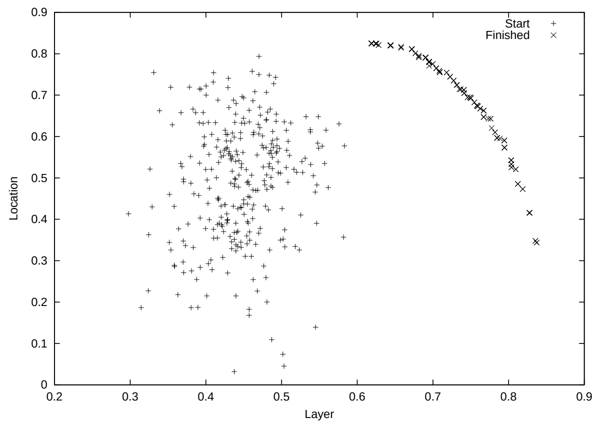
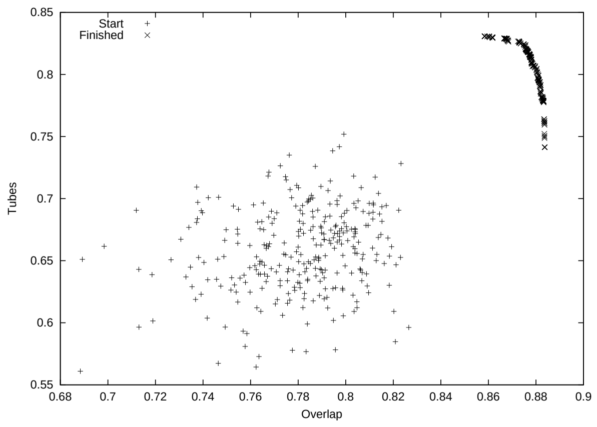

(b) Optimized plant design

Figure 4: 3D plots of the orginal plant (left) and the optimized version (right).



(a) Layer vs. Location



(b) Tubes vs. Overlap

Figure 5: Pareto fronts for four of the criteria and the initial start population.

layout using genetic algorithms. *Computers and Chemical Engineering*, 22:S993–S996, 1998. Supplement 1.

[5] C. A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):129–156, 1999.

[6] J. P. Cohoon and W. D. Paris. Genetic placement. In *Proceedings of the IEEE International Conference On Computer-Aided Design*, pages 422–425, 1986.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: Nsga-ii. Technical report, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology, 2002.

[8] W. E. Donath. Complexity theory and design automation. In *Proceedings of the 17th Conference On Design Automation*, pages 412–419, 1980.

[9] D. Dubois and H. Prade. An introduction to possibilistic and fuzzy logics. In *Readings in Uncertain Reasoning*, chapter IX, pages 742–762. Morgan Kaufmann, 1990.

[10] K. Eguchi, J. Suzuki, S. Yamane, and K. Oshima. *An Application of Genetic Algorithms to Floorplanning of VLSI.* Number 1424 in Lecture Notes in Computer Science. Springer, 1998.

[11] H. Esbensen. A genetic algorithm for macro cell placement. In *Proceedings of the European Desing Automation Conference*, pages 52–57, 1992.

[12] J. P. Evans and R. E. Steuer. A revised simplex method for linear multiple objective programs. *Mathematical Programming*, 5:375–377, 1973.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability – A Guide to NP-Completeness.* International Computer Science Series. Freeman, 1979.

[14] M. C. Georgiadis, G. Schilling, G. E. Rotstein, and S. Macchietto. A general mathematical programming approach for process plant layout. *Computers and Chemical Engineering*, 7(23):823–840, 1999.

[15] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[16] J. H. Holland. *Adaptation in Natural and Artifcial Systems.* MIT Press, 1992.

[17] H. Köpcke and A. Schröder. Constraint Programming versus Logik. Vergleich zweier Ansätze zur Aufstellungsplanung von Chemieanlagen. Technical report, SFB 531, Universität Dortmund, 2004. CI-183/04.

[18] D. Köster. *Ein Assistenzsystem zur methodischen Unterstützung der Aufstellungsplanung von Chemieanlagen.* PhD thesis, Fachbereich Chemietechnik, Universität Dortmund, 1998.

[19] A. Kuziak and S. S. Heragu. The facility layout problem. *European Journal of Operational Research*, 29:229–251, 1987.

[20] C. Y. Lee. An algorithm for path connections and its applications. *IEEE Transactions On Electronic Computers*, 10(2):346–365, 1961.

[21] P. Leuders. *Rechnergestützte Optimierung der Layoutplanung von Chemieanlagen.* Shaker Verlag, 2002.

[22] R. Malingriaux, K.-R. Hilbring, and L. Schuart. Zur optimalen Anordung der Elemente in Anlagen der stoffumwandelnden Industrie. *Wissenschaftliche Zeitung der Technischen Hochschule Otto von Guericke*, 8(14), 1970.

[23] R. Meller and K. Gau. The Facility Layout Problem: Recent and Emerging Trends and Perspectives. *Journal of Manufacturing Systems*, 15:351–366, 1996.

[24] I. Mierswa and T. Geisbe. Multikriterielle evolutionäre Aufstellungsoptimierung von Chemieanlagen unter Beachtung gewichteter Designregeln. Technical report, SFB 531, Universität Dortmund, 2004. Reihe CI.

[25] F. D. Penteado and A. R. Ciric. An MINLP approach for safe process plant layout. *Industrial and Engineering Chemistry Research*, 4(35):1354–1361, 1996.

[26] M. S. Peters, K. D. Timmerhaus, and R. E. West. *Plant Design and Economics For Chemical Engineers.* McGraw-Hill, 2003.

[27] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog, 1973.

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

[29] K. Shahookar and P. Mazumder. VLSI cell placement techniques. *ACM Computing Surveys*, 2(23):143–220, 1991.

[30] H. Tansri and K. C. Chan. A quantitative approach to the plant layout problem using genetic algorithms. In *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 93. Proceedings of the Sixth International Conference.* Gordon and Breach, 1993.

[31] W. X. Yan, C. L. Ning, Y. J. W. X. Yan, C. L. Ning, and Y. Jian. Application of genetic algorithms in plant layout. *Industrial Engineering and Management*, 4(7):30–34, 2002.

[32] P. L. Yu and M. Zeleny. The set of all nondominated solutions in linear cases and a multicriteria Simplex method. *Journal of Mathematical Analysis and Applications*, 49:430–468, 1975.

[33] Q. Yu, S. Badida, and N. Sherwani. Algorithmic aspects of three dimensional mcm routing. In *DAC '94: Proceedings of the 31st annual conference on Design automation*, pages 397–401. ACM Press, 1994.

[34] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.