(R)EVOLUTION IN LAYOUT-PLANNING A NEW HYBRID GENETIC ALGORITHM TO GENERATE OPTIMAL AISLES IN A FACILITY LAYOUT

Knut Alicke, Dieter Arnold, Volker Dörrsam
Institut für Fördertechnik und Logistiksysteme, Universität Karlsruhe,
Hertzstr. 16, 76187 Karlsruhe, Germany
e-mail: [knut.alicke|volker.doerrsam]@mach.uni-karlsruhe.de

Abstract

In the desgin-process of a facility layout, the focus has been mainly on the placement of the machine resources. The material-flow, and therefore the desgin of the aisles has not appropriately been taken into account. We developed a hybrid genetic algorithm (GA) to generate the aisles in a facility layout at minimal fixed and variable cost. The application of the GA to a practical problem leads to remarkable results.

1 Introduction

In todays fast changing markets it becomes more important for a company to compete with low production cost and high flexibility. In the past, the optimization of production- and assembly-lines had been a main subject to lower costs and to increase throughput and flexibility. Production planning and control system were (and will be) developed to meet this demand.

Besides optimizing utilization of the machine resources (production), the facility layout and the material flow are taken more and more into consideration. This is obvious since long and mid-term decisions dominate flexibility and cost optimal production.

Nowadays, planning is done simultaneously, the data are subject to change and detailed during the design process of a facility layout. Common layout planning algorithms neglect the importance of the design of a good material-handling system (MHS) during the stage of the placement of facilities (CRAFT, CORELAP etc.). (Herrmann et al., 1995) and (Kühnle et al., 1996) stresses that the optimization of aisles and network of aisles describing transportation-routes becomes more important in facility design.

In the following section we introduce some specific material-handling terms used in facility-planning. In section 3 and give a mathematical formulation of the objective function of our model. In section 4 we describe the determination of the search space. Section 5 shows how we applied the hybrid genetic algorithm to the problem of generating optimal aisles. The results are presented in section 6.

2 Terms

A production facility as shown in figure 1 can be described with

- *Resources*, which are typically machines like drilling, milling, but also storages. Resources usually contain a local buffer, a pick-up and drop-location and when not operated automatically, the operation-space for the worker. The machine resource can be described by geometrical information, orientation, the position of the pick-up/drop-location and technical information, like service rate, reliability etc.

- *Material flow* between resources can be described with the number of units transported from a source to a destination. By summarizing loaded and unloaded transport moves the transportation flow matrix F can be calculated. Each transport-relation k starts at a pickup-location (output-buffer) of resource i and ends at the drop-location (input-buffer) of resource j. The frequency of transport k is described by $f_k = f_{<i,j>} \, \forall f_k > 0$.

- *Aisles* on which the transport can be done are described by geometric information, capacity restrictions and whether the transport can be done directed (roller or chain conveyor) or undirected (AGV, forklift). Every aisle A consists of a set of aisle-segments a. For every aisle segment an upper capacity bound γ_a (depending on velocity, goods to transport, used containers) can be defined. The space used by a is described by the length and width, $s(a) = l(a) * w(a)$.

In the presented approach, we neglect the upper capacity bound of the aisles-segments and define $\gamma_a = \infty \; \forall \; a$. We further state that the flow on all aisle-segments can be bidirectional and is independent of all other flows. Priority rules at intersections are neglected, so that every transport can be realized at the time it occurs.

3 Objective function

The total cost for a material handling network within a facility layout can generally be devided into a fixed and a variable part.

- *Fixed*: Costs for installation, maintainance, e.g. cost/unit for an AGV or a chain-conveyor and space cost. It is described by the total space needed for all aisles A_s of the facility layout under consideration:

$$C_f = S(A_s) = \sum_{a \in A_s} s(a) \tag{1}$$

- *Variable*: Performing transports results in running (variable) cost for power, service, auxiliary material and personal. Additionally the transportation time has a serious impact on investments. The longer the overall flow the more transporters are needed to fulfill the transportation task. The variable cost can be calculated based on the length of each aisle segement A_k of the material flow k and the corresponding intensity f_k:

$$C_v = \sum_k S(A_k) = \sum_k f_k \sum_{a \in A_k} s(a) \tag{2}$$

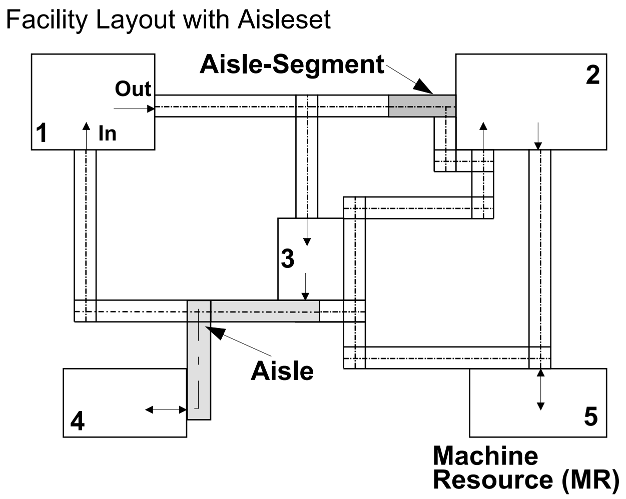

Figure 1: Terms related to material-handling

To minimize C_v of a given layout, the path-length for every aisle A_k has to be minimized. This can be achieved by using a shortest-path algorithm but leads to a layout with a large number of aisles, each only used for one transport-relation, therefore high fixed costs C_f.

To minimize C_f on the other hand, as many aisle-segments as possible should be used multiply. This can be obtained by generating a Minimal Steiner Tree (MST). But a MST can lead to detours, therefore high variable cost C_v.

Another important objective is to minimize the number of turns t in the aisle-layout. They are necessary for a compact facility layout, but cause rising transportation time (impact on variable cost). Moreover, every turn has to be realized with expensive divert elements (impact on fixed cost). The total cost for turns in an aisle-set is denoted by C_t.

With a given facility layout, the objective function consists of the three (conflicting) terms as follows, where g_1, g_2 and g_3 are weighting factors:

$$MIN \quad Z = g_1 C_f + g_2 C_v + g_3 C_t = g_1 \sum_{a \in A_s} s(a) + g_2 \sum_k f_k \sum_{a \in A_k} s(a) + g_3 t(a) \tag{3}$$

The problem of generating an optimal material-flow network is related to problems from telecommunication, computer and teleprocessing networks. An introduction and survey into designing optimal networks is given by (Minoux, 1989). (Herrmann et al., 1995) describes the problem of generating optimal aisles in a facility layout as a fixed charge network design problem and presents two grid-based heuristics to solve it.

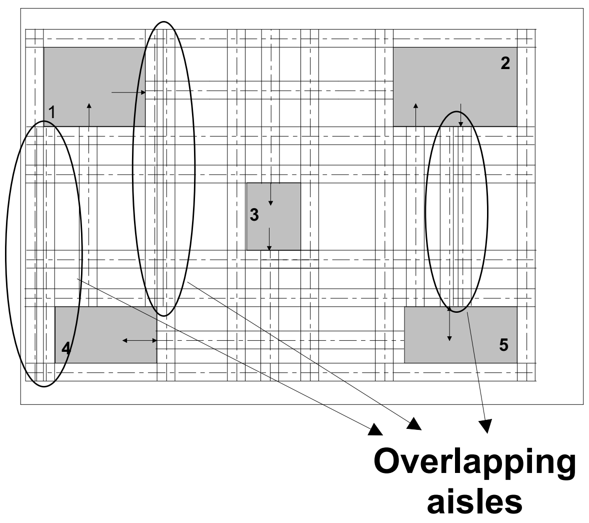

Figure 2: possible aisle-segemnts (full search space)

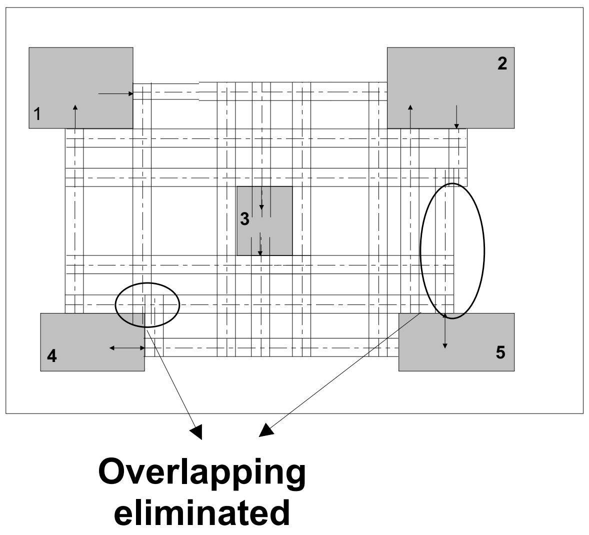

Figure 3: possible aisle-segments (reduced search space)

4 Defining the search space

As mentioned, minimizing C_f can be identified with being a Steiner-Problem. (Ganley and Cohoon, 1993) describe the generation of an *Escape Graph* to create the solution space for the rectilinear Steiner Tree Routing in IC design. For our problem this generation method is very useful if we identify aisle segments in the layout problem being escape segments in IC design. It provides the optimal solution for the steiner-problem *and* for the second term in the objective function. The proove can be found in (Ganley and Cohoon, 1993).

The method presented by Ganley and Cohoon is extended by a heuristic to ensure nonoverlapping aisle-segments. Two steps are needed to create the full search space. Then the search space is reduced by some pre-processor techniques.

First horizontal and vertical sweeps of boundary segments of both obstacles and production facilities are created. For horizontal sweeps, the segments are drawn half the width of an aisle to the north of a boundary segment if the boundary segment represents an upper contour, and half the width to the south if it represents a lower contour accordingly. Each segment ends half the width of an aisle before a vertical boundary segment or the boundary of the whole facility. Vertical sweeps are performed in an analogous manner.

Second, each drop/pick-up location of each production facility induces one escape segment running vertically to the appropiate boundary segment from a drop/pick-up location of a production facility. Time complexity of the generation of the whole graph is $O(\max n, m \log m)$ where n is $O(m^2)$ in the worst case (Ganley and Cohoon, 1993).

In contrast to the common grid-based approach for designing a useful transportation network (Herrmann et al., 1995) the algorithm described here ensures that the search space contains the optimal solution. Furthermore it enables the user to use any given CAD based layout. Figure 2 shows the full serach space.

4.1 Reducing the number of aisles

As described in (Ganley and Cohoon, 1993) the escape graph can be reduced dramatically by deleting unnecessary segments. By using a shortest-path test, all vertices which are not on a shortest-path connection for any source-destination combination can be deleted. Secondly Yang and Wing (Yang and Wing, 1972) describe that a vertex being a corner vertex can be eliminated if it is not a drop/pick-up location. Finally every other vertex with two edges can be eliminated if it is not a drop/pick-up location.

Besides these eliminations still a complex graph exist. Any further reduction technique is based on heuristic assumptions which are implemented to tackle the third cost term C_t. The segments of an *escape graph* do not have any space requirements but aisles segements do have space requirements. Unnecessary corners can be avoided by eliminating segments which overlap with other segments and induce corners at the ending points. A straight line of segments is dominated by another line of segments if the second line forms a longer sweep.

Therefore the dominated line can be deleted. If two lines have the same length an arbitrary node of one line can be choosen, to devide the two overlapping lines in such a way that no overlapping of any aisle segment occurs.

It has to be mentioned that optimality is only guaranteed for non-capacitated flows on the aisle segments. A reduced serach space after applying the heuristic elimination technique is shown in figure 3.

5 Hybrid Genetic Algorithm

Based on the same idea of adapting the optimization procedure of the evolution to mathematical problems, genetic algorithms (GA) and evolutionary strategies (ES) differ in the realization. For our application we implemented and algorithm consisting of both GA and ES, therefore we called it an hybrid genetic algorithm.

5.1 Coding-Datastructure

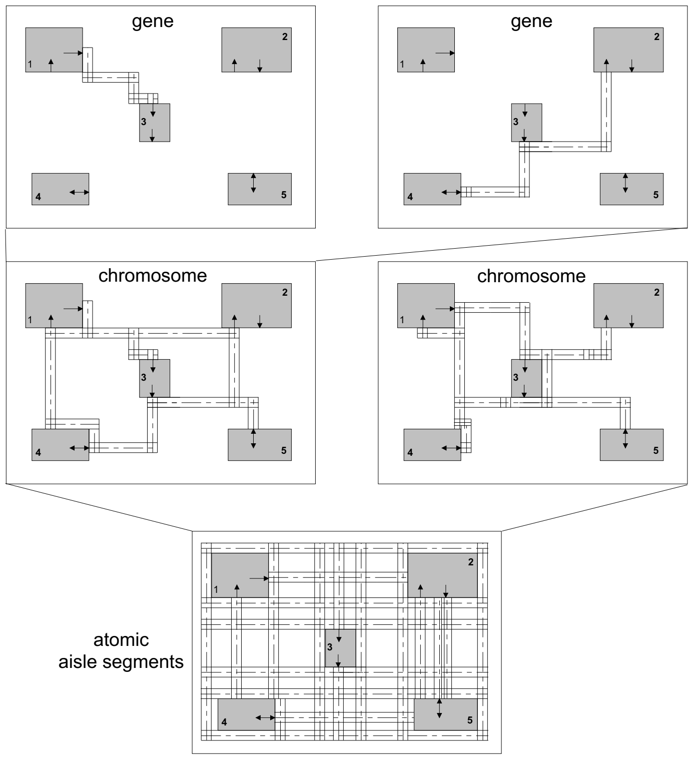

Figure 4: Hierarchy of objects

The *genetic coding* should allow an easy application of standard-operators like the one-point crossover. On the other hand, it should be resource-saving and effective. We developed objects for a gene, a chromosome and a population. These objects base on the set of possible *atomic* aisle-segments, computed as described in section 4 and shown in figure 2 and figure 3. The hierarchy of the objects is shown in figure 4. Pointers are used to have a consistent and resource-saving structure. Therefore the set of possible aisle-segments has to be stored only once. The objects contain the data-structure and operations to calculate the fitness, etc. The data-structure is seperated from the operations.

The central issue in our application is to ensure the transport between two machine resources on a given aisle. So we defined a gene as an aisle, each gene containing a set of pointers to aisle-segments. One aisle-segment can belong to more than one gene.

A chromosome represents one valid solution for the problem, for each connection of the material flow matrix one gene exists in the chromosome. All chromosomes have the same size and contain again a set of pointers to the corresponding genes.

The algorithm operates with one parent and one offspring population, P_p and P_o respectively, with a fixed number of chromosomes. The functionality, the genetic operators, fitness-function, selection and recombination, are implemented in the objects as described below.

5.2 Fitness

The calculation of the fitness is done according to the described objective function (3) and implemented as an operation of the chromosomes. To be able to compare the chromosomes, the terms of the objective fuction are normalized with the length of the minimal path of each aisle $l(min\_Path_k)$ and an approach of the length of the MST $l(MST)$. The minimal number of turns for m transport-relations is $2m$. Therefore the fitness function is:

$$MIN \quad Z = \frac{1}{g_1 g_2 g_3}\left(g_1 \frac{l(MST)}{C_f} + g_2 \frac{\sum_k l(min\_Path_k)}{C_v} + g_3 \frac{2m}{C_t}\right) \tag{4}$$

To calculate the value of the first term, the total length needed by the aisles, one temporary gene is generated which contains every aisle-segment of the genes of the chromosome. The calculation of the second term, the length of each aisle is done by summarizing the length of each aisle-segment of a gene. Than, the length of the aisle-segments is summarized.

5.3 Initialisation

The inital chromosomes should be different from each other and widly spread over the solution space. To obtain this, we select a random aisle-segment a and generate the genes of the chromosom such that they contain the selected aisle-segment. That means, that all aisles of one chromosome contain the aisle-segment a, which is definitely not a good solution to the problem. But every chromosome is initialised using an other aisle-segment, so after a few generations the chromosomes are still widly spread over the solution space. Evidently, in this phase the increase in fitness is high.

5.4 Selection

The higher the fitness of the chromosomes, the higher is the probability to be chosen as parent. A common used operator is *ranking*. The chromosomes are sorted by their fitness and the probability to be chosen is computed according to a given distribution. We used a linear degressive distribution to compute this probability.

5.5 Crossover

The crossover-operator is used to obtain new genetic material, in our application new or changed aisles, respectively genes. Two *parents* are combined to one or more *offsprings*. We applied the standard operator, *one-point*-crossover. With the used coding, it is ensured, that a valid solution is obtained.

It is possible, that the offspring contains aisle-segments, which are actually not used. But perhaps they are part of a scheme (Schöneburg et al., 1994) which becomes more important at a later generation. Because all costs are higher than without the aisle-segments, the fitness of the chromosome is worse.

This could be the aisle-segments of a detour in the actual offspring, never used to meet the transport-demand. In a later recombination, the aisles of this detour could perhaps be a shortcut or multiply used. These *unnecessary* aisle-segments are kept in the chromosome, but we implemented a mutation-operator to delete them. Otherwise, the chromosomes would gradually be filled up with unused aisle-segments, the fitness would become worse and worse, and the algorithm would degenerate.

5.6 Mutation

The genetic operator *mutation* should avoid the degeneration of a population. Degeneration means that the algorithm stucks on a local sub-optimum. Parts of a chromosome are modified in a random manner to introduce new genetic material.

The *first* mutation-operator deletes unused aisle-segments. As mentioned before, it is important to handle the unused aisle-segments which are perhaps introduced during the crossover. Based on the given chromosome, the shortest path for each gene is identified. These are stored in one temporary gene. Than it's checked which of the aisle-segments of the chromosome are not in the gene, these are deleted.

The fitness of the chromosome is higher or equal than before, therefore we also call this operator a local optimizer. But keep in mind, that these aisle-segments are deleted and cannot be used in a later generation for other purposes.

The *second* operator really introduces new genetic material. Here, one random gene is chosen and the existing aisle is replaced by the one representing the shortest path. According to the length of the path and the fact that we are using a rectlinear matrix, the shortest path is not definite. We consider additionally the number of turns on the way, which reduces the number of possible shortest path. Out of the remaining, we choose the one, where most of the aisle-segments are already used by other genes of the chromosome. Therefore, this operator is also a local optimizer, because it tries to find an aisle, which gains the most fitness of the whole chromosome. Aisle-segments are introduced and deleted.

The *third* operator introduces new aisle-segments, without respect to the fitness. A gene out of a chromosome is chosen randomly and than deleted. One random atomic aisle-segments is chosen and the pick-up and drop-location of the deleted gene are connected on the shortest path to this aisle-segment.

It is obvious, that at the beginning of the optimization process, the most radical mutation operator, the third one, should be applied with a higher probability. The probability of the first two should be increased at the end of the run. Therefore we implemented linear de/increasing probability functions for choosing the weight of the mutation operators.

5.7 Recombination

After the application of the crossover and mutation-operators the new chromosomes are generated. We applied the *survival of the fittest*-rule to obtain a new population. The chomosomes of both parent and offspring-population are sorted by their fitness and the n best are copied into the population of the new generation.

6 Results

The problem of generating aisles in a facility layout leads to an objective function with conlicting terms. We developed a coding such that the application of standard crossover-operators of GA and ES are possible. The mutation-operator was developed both as local optimizer and to introduce new genetic material.

One advantage of the GAs to other methods is its implicit parallelism. The best chromosomes of the population have nearly the same fitness, but the layout of the aisles can be different. So the planner can chose one of the *near-otimal* solutions, or approach the desired solution by an interactive process with the GA.

In a practical project, we had to redesign a facility layout with 62 machine resources. This problem was solved in reasonable time.

7 Further Research

For practical application, it is important, that aisle or aisle-segments can be fixed during the optimization process that capacity bounds and the direction of the used conveyor elements are considered. In this first step, we did not want to restrict the GA more than necessary. But the practical requests can easily be achieved by a slight modification of the initialisation and the mutation-operators.

8 Conclusions

The problem of generating aisles in a facility layout leads to a objective function with conflicts and is NP-hard. We used a combination of GA and ES to solve the problem. We developed a coding such that standard-operators could be applied. So the algorithm runs fast and saves resources.

A problem with 62 machine resources was solved in reasonable time and due to the implicit parallelism of the GA, the planner can chose out of different solutions, equal in the fitness. The results are remarkable.

References

Ganley, J. L. and Cohoon, J. P. (1993). Optimal rectilinear steiner tree routing in the presence of obstacles. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.*

Herrmann, J. W., Ioannou, G., Minis, I., Nagi, R., and Proth, J. M. (1995). Design of material flow networks in manufacturing facilities.

Kühnle, H., Fietz, R., and Schmidt, C. (1996). Factotum - ein entwicklungsfähiges Instrument zur computerunterstützten Produktionssystemplanung. *Industrie Management*, 12(3).

Minoux, M. (1989). Network synthesis and optimum network design problems: Models, solution, methods and application. *Network*, 19.

Nissen, V. (1994). *Evolutionäre Algorithmen - Darstellungen, Beispiele, betriebswirtschaftliche Anwendungsmöglichkeiten.* PhD thesis, University of Göttingen.

Rechenberg, I. (1973). *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evoultion.* Frommann-Holzboog.

Schöneburg, E., Heinzmann, F., and Feddersen, S. (1994). *Genetische Algorithmen und Evolutionsstrategien.* Addison-Wesley.

Yang, Y. Y. and Wing, O. (1972). Optimal and suboptimal solution algorithms for the wiring problem. *Proceedings of the IEEE International Symposium on Circuit Theory.*