Optimization of 4D Process Planning using Genetic Algorithms

F, Märki, M.A. Suter, M. Vogel and M. Breit
AEC Informatics Institute, University of Applied Sciences North–Western Switzerland
Steinackerstrasse 5, CH–5210 Windisch, Switzerland

Contact: m.vogel@fh-aargau.ch

Summary
The presented work focuses on the presentation of a discrete event simulator which can be used for automated sequencing and optimization of building processes. The sequencing is based on the commonly used component–activity–resource relations taking structural and process constraints into account. For the optimization a genetic algorithm approach was developed, implemented and successfully applied to several real life steel constructions. In this contribution we discuss the application of the discrete event simulator including its optimization capabilities on a 4D process model of a steel structure of an automobile recycling facility.

1 Introduction

1.1 4D modeling of building processes
The work described in this paper is part of an ongoing R&D project on 4D modeling (in space and time) of building processes. The project is a co-operation of the AEC Informatics Institute (AEC-II) of the University of Applied Sciences North–Western Switzerland, the Center of Integrated Facility Engineering (CIFE) at Stanford University and industry partners in Greece, USA, Germany and Switzerland. The objective of the program is to develop a software prototype for the modeling, visualization, analysis, optimization and evaluation of building processes in the context of the whole life-cycle of a building. 3D computer models of architects and engineers will be linked with process information and visualized over the time. Such 4D models will allow to simulate risk scenarios and support the decision making process and the optimization of the planning.

1.2 Set-up for the development of genetic algorithm
The starting point of the work was the Construction Method Modeler CMM (Aalami, Kunz, Fischer 1998) written in Power Model, an AI development tool in the UNIX environment, which is only portable to a very limited degree to other operating systems. The use of 4D models in practice will strongly depend on the quality of graphical user interface and the response time of the application to provide the appropriate visual and structural information for complex process design problems. For this reason the further development at the AEC-II uses the NURBS based Java-3D environment C3DQ from CCT in Athens as graphical user interface, which is linked to a SQL database. In order to allow parallel development of future components of the 4D environment the development of the genetic algorithm for process optimization has been set up as a modular application GAPO (genetic algorithm process optimization) which runs independent of the 4D environment. The necessary information flow is established with XML data files. The test setting for GAPO (see Figure 1) uses Bocad-3D – a steel detailing package – for 3D modeling. Rinoceros – NURBS modeling for windows – is used for the conversion of 3D data into VRML. The additional data editing was carried out with VrmlPad from ParallelGrafics. GAPO does the sequencing, scheduling and optimization of process plans and Invizn from Stanford and Disney serves as 4D Viewer. GAPO can also export the process

plans to MS Project. The Invizn 4D animations can also be exported to 4D-VRML with the tool 4DGEN developed at the AEC-II (Raps 2002).

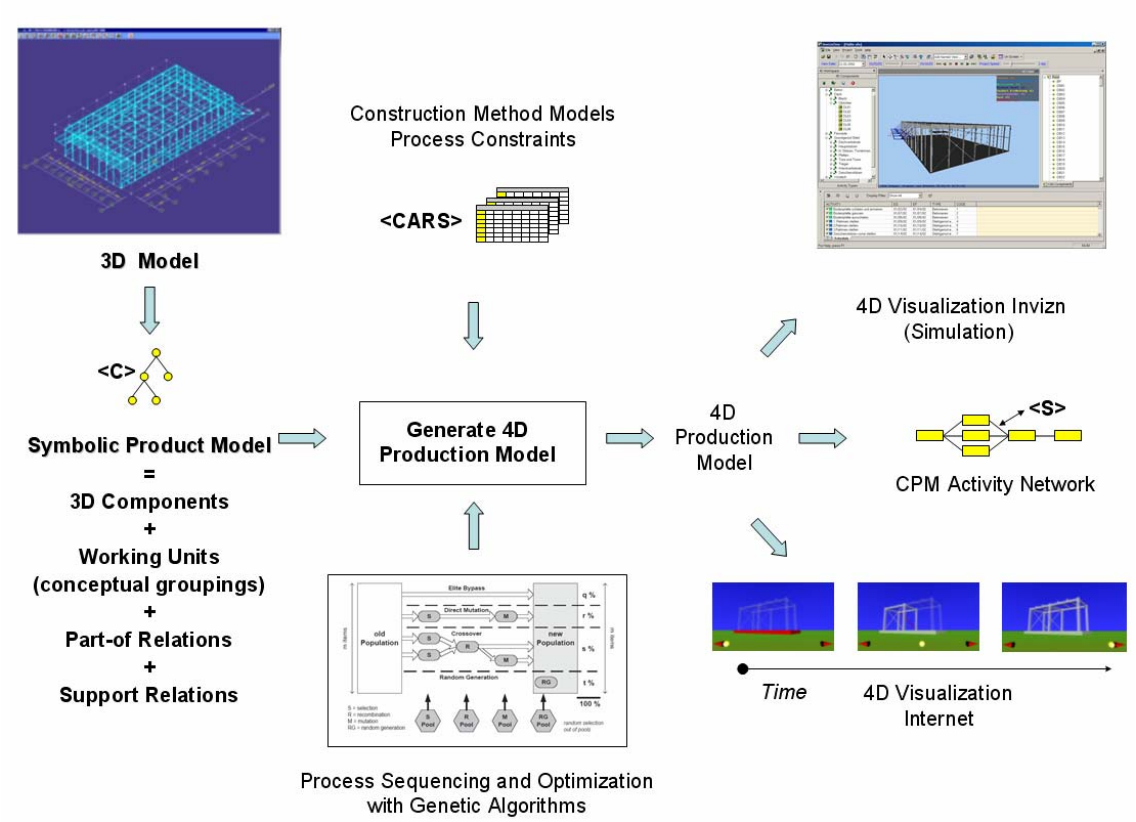

Figure 1: Information flow in the GAPO development environment

2 Symbolic product and process model

2.1 From a CAD model to a product model

3D models from architects and engineers are the basis for 4D modeling. Usually those models provide the geometric description of the components. The minimum structural information needed is named objects. For 4D process modeling, the structure of the building and the relations between components or groups of components has to be added to the model. We use hierarchical structures. An example is shown in Figure 2. Typical structural classes are buildings, areas, floors, conceptual groupings (assemblies), etc. These objects have relations such as the "part-of" or the "support" relation. The latter is necessary to decide, whether or not a component can be built in the structure. We have done research and development to support the user in structuring 3D models and to propagate relations (Mettler and Krauer 1999). For this work we only use the support relation. Bocad-3D provides rich structural and geometrical information to support the building of a product model. The 3D model is completely structured and the connections between components are fully modeled. The differentiation between types of connections into shop or field connections helps to create conceptual groupings of elements (Mettler and Krauer 1999). Presently this information can not directly be exported. Thus, a certain amount of manual editing is necessary.

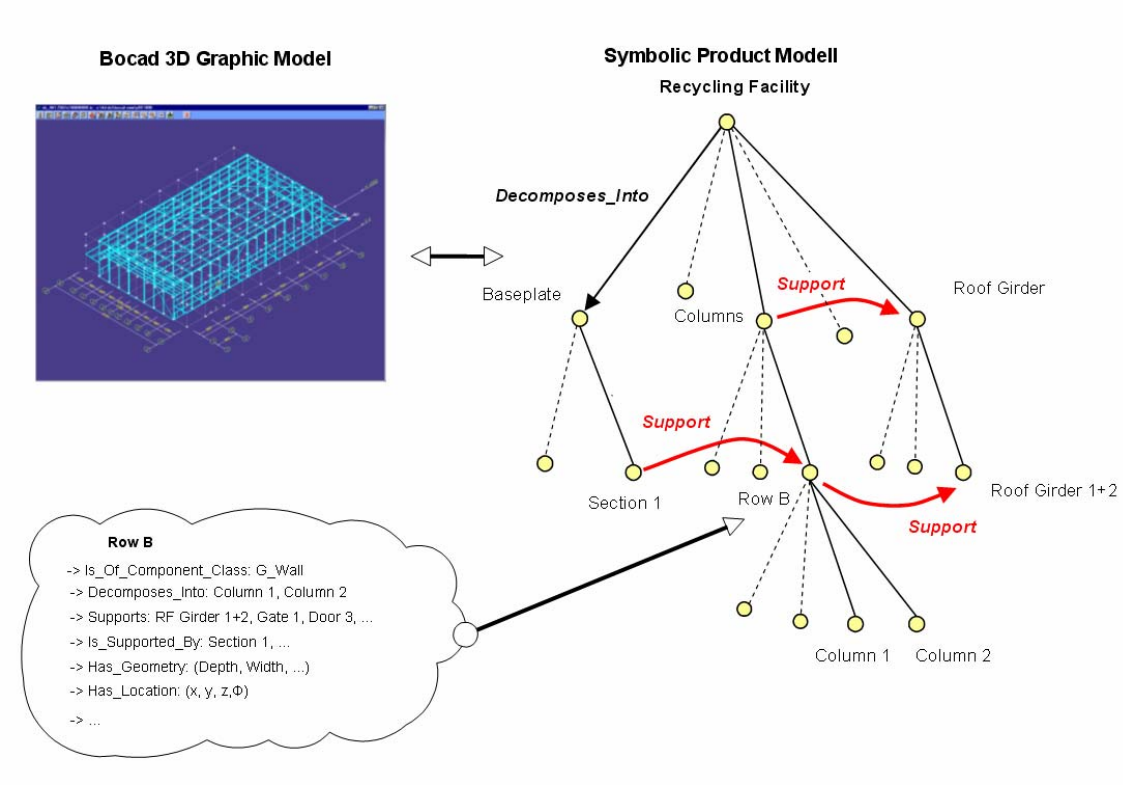

Figure 2: Product model of the recycling facility

2.2 Data structure

In the product model the component structure and the relations between them are described. For the process model activities and resources have to be added.

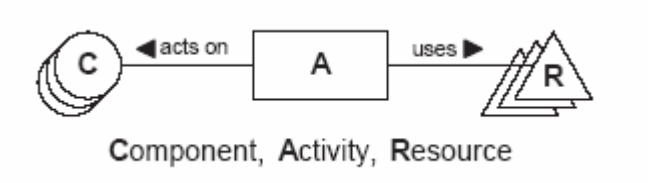

Figure 3: <C,A,R> –Tupel

The data structure uses <C,A,R> –Tupels (Aalami, Kunz, Fischer 1998), with the activity being the central element. The activity is assigned to a number of components it acts on and uses resources. Components are 3D objects as well as hierarchical structures of conceptual groupings, which are related to each other by the support relation. The activity as part of the net plan has a predecessor and a successor. Activities may have process dependent constraints, which regulate their sequence to one another. Resources are workforces, used systems and materials.

2.3 Sequencing and scheduling

For a given product model professionals define the necessary activities and assign the types of resources needed to perform them. The challenge is to order the activities in a way that the building processes will be performed in an economical manner. This is done in two steps. First, all activities will be sequenced based on the given structural and process related dependencies. This leads to a net plan which complies with all constraints, but neglects the resource availability. In the second phase – called scheduling – the balancing of resources is performed, which changes the duration and in some cases the sequence of activities in the process plan. The scheduling part is done by a genetic algorithm approach which is described in the following chapter.

2.4 Discrete event simulator for sequencing of activities

For the sequencing of activities we chose a direct approach which uses the hierarchical structure of the product model represented as a tree, where the leaves are geometrical components. The discrete event simulator sequences the activities in determining if the needed support of a component is given and if a process related constraint allows that the activity can be performed. The following listing shows the pseudo code for this sequencing. In the given example the activity "set frame 2" acts on frame 2, which consists of two columns and a beam. The erection of frame 2 can only be carried out, if the appropriate anchors have been placed.

```
sequencing {
  while(not all activities sequenced) {
    choose an unsequenced activity x
    set A = all components x acts on
    set B = all leaf components of A
    set C = all father components of A
    set D = (all supported by components of B) \ B
    set E = (all father components of D) \ C
    set F = activities assigned to E
    elements of F are predecessors of the processed activity
  }
}

processed activity = [set frame 2]
set A = [frame 2]
set B = [column A2, column B2, beam A2-B2]
set C = [frames, Recycling Hall]
set D = [anchor A2, anchor B2]
set E = [anchors, foundation]
set F = [place anchors]
predecessor of "set frame 2" is "place anchors"
```

Figure 4: Pseudo code and an example for direct sequencing of activities

3 Genetic algorithm approach

3.1 Genetic algorithms applied in process planning

Genetic algorithms (GA's) are heuristic search methods based on the Darwinian principle of evolution. In living organisms two individuals intermix their genetic material in order to produce offspring which are better adapted to their environment. Similarly, genetic algorithms start from a pool of objects which are scored by a fitness function measuring their quality as a candidate solution of a given problem. By some probabilistic mechanism these candidate

solutions are exposed to a kind of artificial evolution consisting of selection, recombination and mutation yielding a new generation of candidate solutions which are expected to have a higher quality or fitness. Genetic algorithms have been successfully applied to a wide variety of practical problems in diverse fields like chemistry, biology, operations research and many engineering disciplines. Since we are not aware of any publications attempting to use GA's for optimizing process planning, we briefly summarize our approach. A more detailed description will be give elsewhere.

In the following we distinguish between sequenced activities (net plans) and project plans:

Net plans: sequenced activities scheduled with predecessor relationship
Project plans: net plans scheduled with time and resource management

A serious obstacle for applying GA's on project plans is that some GA-operators might violate some scheduling constraints. The main challenge was therefore to find a suitable data structure for encoding the "genetic" information of process plans which allows an efficient evolution but conserves all sequencing constraints.

Our GA–model starts with an initial population of randomly generated project plans. The number of individuals in this initial population can be determined by the user. A subsequent population will then be assembled using 4 strategies which can also be weighted by the user. A fraction *q* of the best individuals will be directly passed to the next population. This guarantees that the quality of the most suited candidates will monotonically increase from generation to generation. A second fraction *r* of individuals will be passed to the next population after a mutation. This process will help to find a local optimum but also opens new search regions. A third fraction *s* of the new population is created by recombining selected individuals of the old generation; some of them undergo a mutation after the recombination. Last, a fraction *t* of the new population is created randomly. In general, the new population has the same size as the old one, but this is not a necessity.

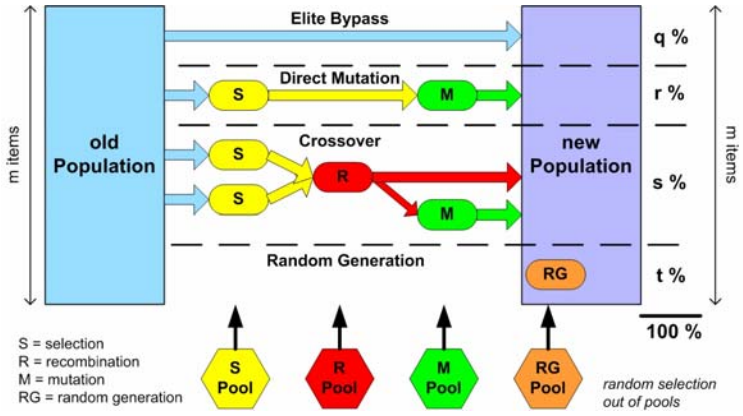

Figure 5: The GA–model

3.2 Data–structure

The success of GA's in optimization problems crucially depends on an appropriate encoding of the different parameters being optimized. In addition, in process plan optimization sequencing constraints have to be satisfied. Our representation of a project plan as a chromosome is shown in Figure 6. The chromosome is defined as a sequence of genes representing the sequencing relation defined by the net plan. All genes differ from each other and have particular characteristics which can be stored in a central collector.

According to the net plan, activities are allocated to positions and to resources as shown in Figure 7. All this information is stored in the genes. In the simple case, an activity can unambiguously be allocated to a well defined position. On the time axis such an activity will be scheduled somewhere between the start and end time of the corresponding position. Within this position the activity can be shifted and is only constrained by the resources it is using. In more complicated cases an activity can be allocated to several positions, such an activity is called a **commuter**. It might be that such a commuter starts in one position and extends over some subsequent positions. In order to conserve all relationships commuters must also be allocated to all possible follow up positions. Again, its particular scheduling on the time axis will be determined by the used resources.

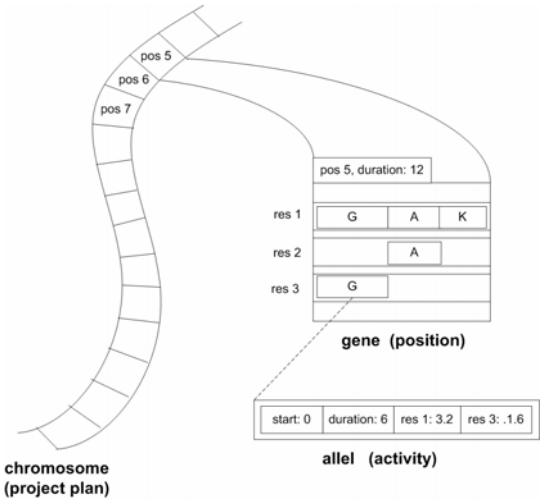

Figure 6: Genetic representation of a project plan

A particular project plan is thus given by a chromosome consisting of a sequence of genes, called positions. Each position contains of allocated activities which are fixed relative to this position by their start times and durations. To each activity a number of resources can be assigned. All activities allocated to a position will define the duration of this position, whereby special attention has to be given to the influence of the commuters.

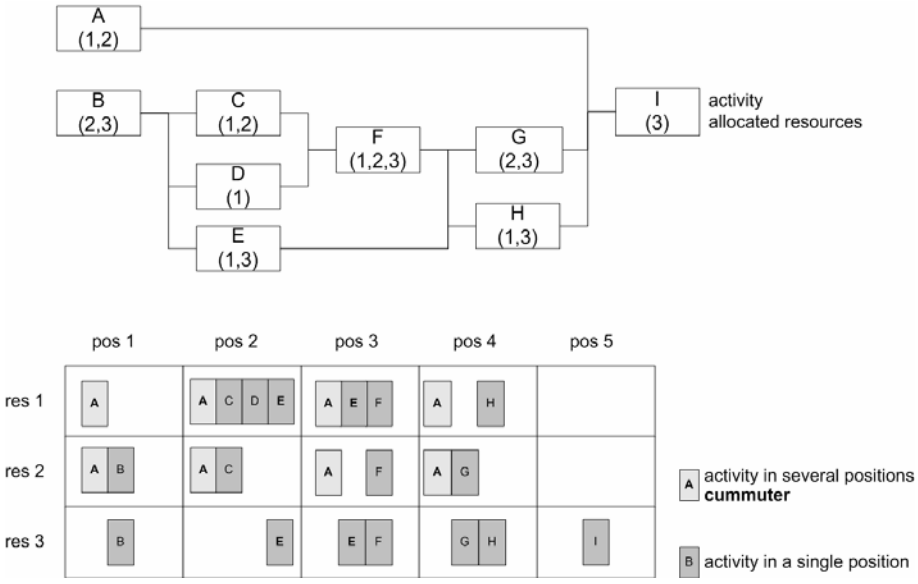

Figure 7: Example of a simple net plan and its allocation to genes (positions)

3.3 Genetic operators

A number of operators are required to manipulate the chromosomes of a given generation in order to construct a new generation of hopefully fitter individuals. In general we distinguish three types of operators: selection, recombination and mutation.

The selection determines who gets to mate which in general depends on the fitness of the individual. We have implemented the following selection operators:

EliteSelector:	selects randomly from the best project plans
RandomSelector:	selects randomly from all project plans
RouletteWheelSelector:	fitness weighted selection from all project plans

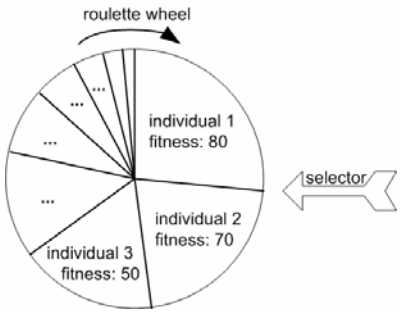

Figure 8: Roulette–wheel selection using fitness weighted probability

During recombination chunks of genetic information are exchanged between a pair of chromosomes. This allows the creation of genetically radically different offspring that are still of the same general flavor. We have implemented the following recombination operators:

SimplePositionRecombinator:	recombines two parent projects by cutting at arbitrary positions
ShorterPositionRecombinator:	selects for every position from two parents the shorter one
ResourceThreadRecombinator:	selects resource threads and fixes possible resource conflicts
ResourceUseRecombinator:	selects activities from one and resources from an other parent

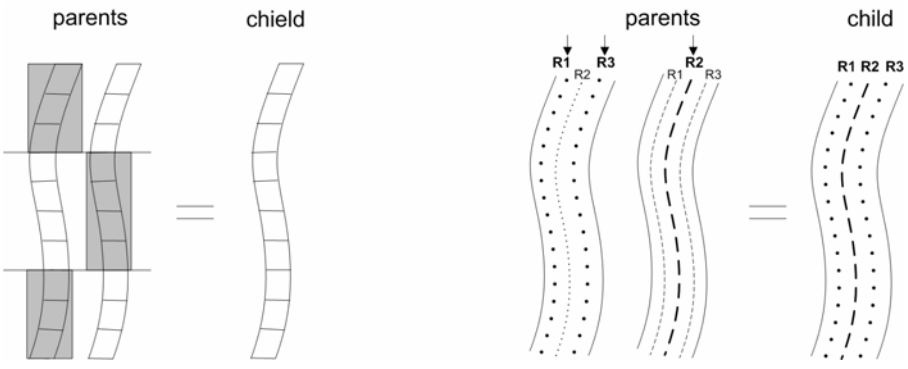

Figure 9: Recombination of positions and resource threads

Mutators operate on activities by either changing their parameters or positions arbitrarily. Possible resource conflicts are automatically resolved. The purpose of a mutation is twofold. It first provides candidates which cannot be obtained by recombination and secondly, it prevents an early convergence at a local extreme by introducing inhomogeneities and diversities into the population. It therefore counterbalances the selection pressure. Mutation might also yield a fine tuning of candidate solutions. We have implemented the following mutators:

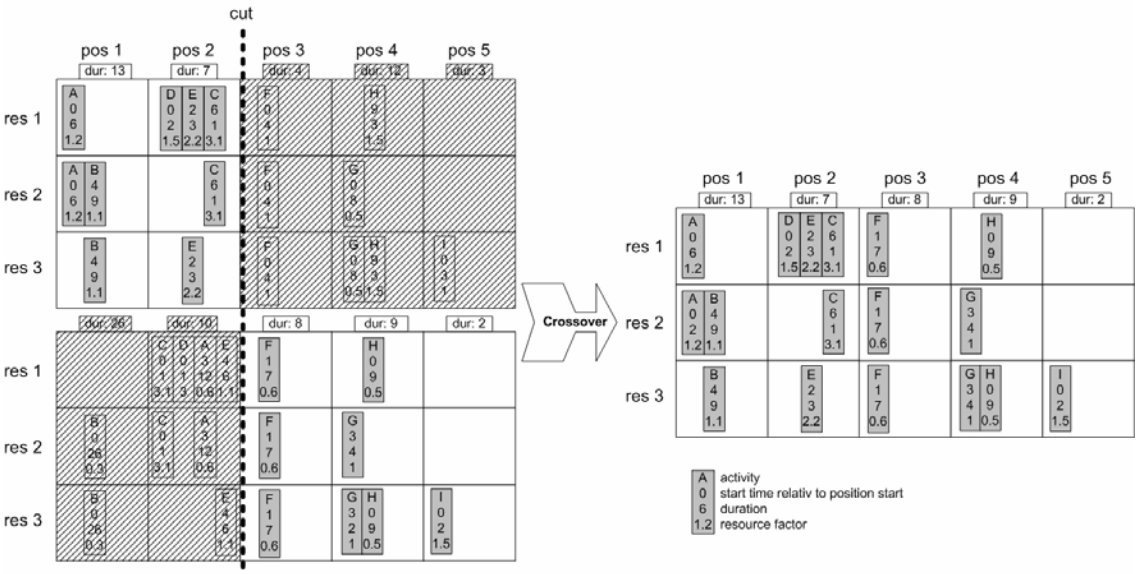

Figure 10: Position recombination with a simple one–point crossover

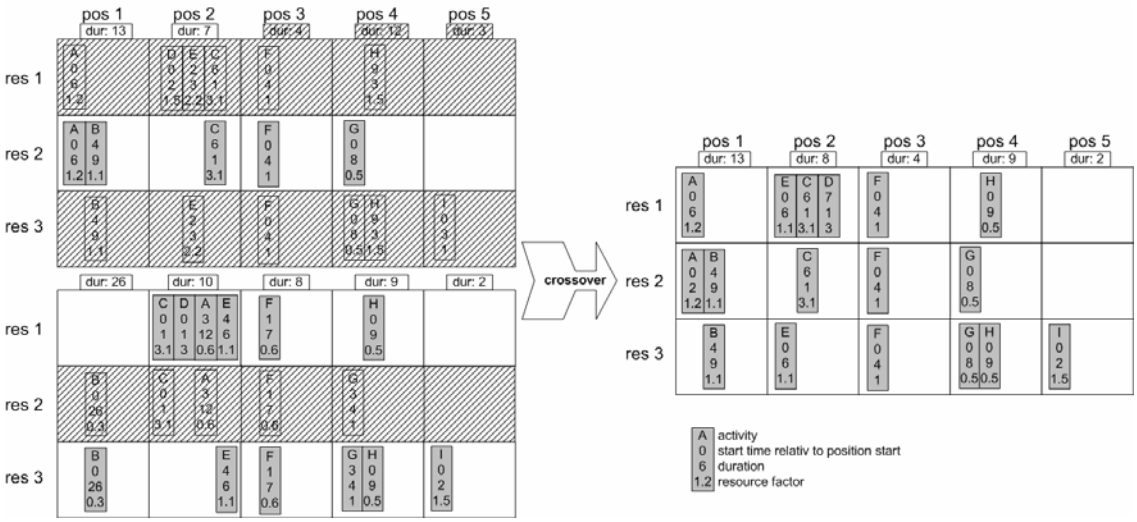

Figure 11: Recombination of resource threads

ResourceExertionMutator:	mutates the allocated resources at some positions/activities
RandomShuffleMutator:	mutates the sequence of activities within a position
PackerMutator:	tries to pack activities as close as possible
CommuterMoveMutator:	mutates the position of commuters

In addition to the above operators we also use a random project plan generator. The very first population is created by this generator but we also use it during the optimization process as already described in our GA–model (see Figure 5). This random generator produces a valid project plan by taking into account all scheduling and resource constraints.

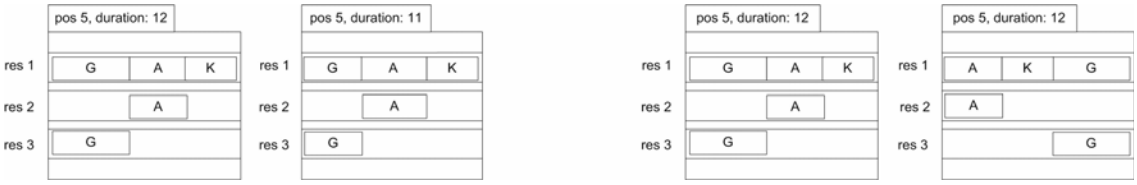

Figure 12: Mutation of resource amounts (left) and activity positions (right)

3.4 Fitness functions

The quality of every project plan is measured with a fitness function. In general a project plan is superior if its duration is shorter or if the corresponding cost is lower. In contradiction to the common convention we therefore use fitness functions assigning smaller real numbers to fitter project plans: the higher the quality of the project plan, the smaller its fitness.

For measuring the fitness of a project plan several aspects can be taken into account: the total project duration, the duration of resource use, the amount of resources, increase, reduction and change of resources and unused resources. These aspects can be rated by the user and provide a final fitness consisting of a weighted mixture of the following fitness functions:

ProjectDurationFitnessFunction :	measures total project duration
ResourceDurationFitnessFunction :	measures duration over which resources are used
AreaFitnessFunction :	integrates amount of resources used during the project
ResourceChangeFitnessFunction :	integrates changes in the amount of used resources
GapFitnessFunction :	measures duration where available resources are unused

Like all the GA-operators, the above fitness functions can be switched on or off by the user in a dialog box allowing any desired combination of them. This provides the possibility to optimize according to individual criteria. New operators or fitness functions can be added dynamically.

3.5 Scaling behaviour

In the next chapter we will show that our genetic algorithm optimization can successfully be applied to real life problems. The main question in this regard is about the storage and CPU requirements. We will therefore briefly address these questions.

In the practical project of the car recycling facility discussed in the next chapter the maximum allocated memory is about 20MB / 32MB / 56MB if a population size of 250 / 500 / 1000 is chosen. This indicates that much bigger projects are tractable with GAPO. If memory problems will occur in a huge project, we can switch to an other representation of project plans since the one used contains quite some redundant information. This redundancy is in favor of the performance and could be avoided by a project representation in two layers. One layer will be stored and contains only the required information to reproduce the detailed characteristics used in layer two for the calculations. The basic information of a project plan stored in layer one are: all activities, the start step of every activity and the amount of every resource used per activity. All the other data used in the calculations could be determined from these informations. We can also save memory by choosing a smaller population size as indicated by the above numbers.

A more serious problem might be the required CPU time for the optimization. Of course this depends strongly on the chosen parameters and on the hardware. We have empirically investigated the complexity of the optimization and we have found that the optimization time is $O(n^2)$, where n denotes the number of activities. The influence of the number of resources on the performance is negligible. As mentioned above, the scaling constant depends on the hardware and on the adopted optimization parameters. To give an indication we give an example of what can be calculated in 10 minutes on a standard 2.8 GHz/Pentium 4 computer: 100 generations with 250 project plans each, with 200 activities and 15 resources. For comparison, the automobile recycling facility discussed in the next chapter has 63 activities and 5 resources and is fully optimized in a few minutes.

4 A practical application

4.1 Facility for the recycling of cars

A one story high steel hall should be erected for a planned recycling center. The owner develops a new prototype for the recycling of cars. As the machine layout is not determined at the time of order the position of internal columns is in questition. For our example we use the final design, which deploys 60 m long single beams (plate girders).

4.2 Design and construction issues

The facility has the following dimension length / width / height = 60 m / 30 m / 10 m. It has a light, hinged structure, stabilized through bracings in the roof and walls. 3D design and detailing was used. Shop connections are welded and field connections are screwed. For construction mobile cranes (25t / 60t / 100t) and hydraulic scissor lifts are available. Steel crews are typically set up with 3 laborers, one to serve the crane, the others to set and fix the structural elements. The erection processes are as follows: frame-wise erection, temporary fixing of bracings and secondary elements; adjustment and final torque.

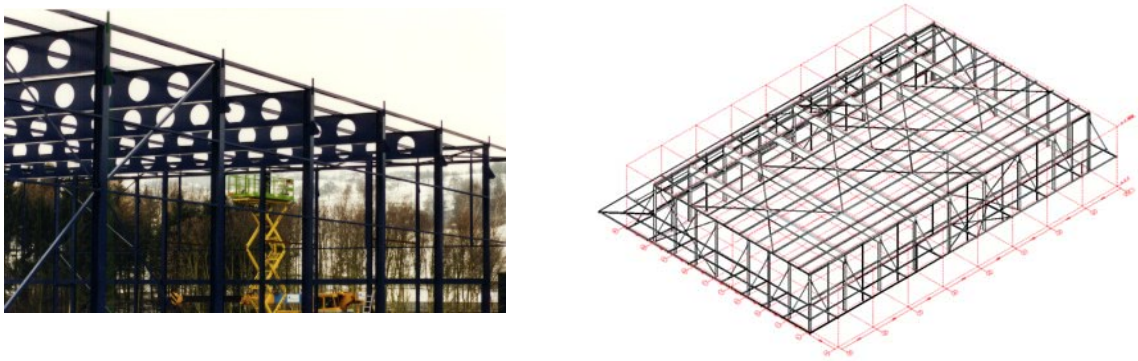

Figure 13: Recycling hall, construction and 3D model

4.3 4D modeling and application of genetic algorithms

Based on the Bocad-3D model a 4D process model was generated as described in the former sections. The model consists of 305 geometrical 3D elements, 23 conceptual groupings, 63 activities and 5 resources (concrete workers, steel workers, unskilled workers, cranes and scissor lifts). After sequencing, the process plan was optimized using genetic algorithms with the fitness functions described in section 3.4.

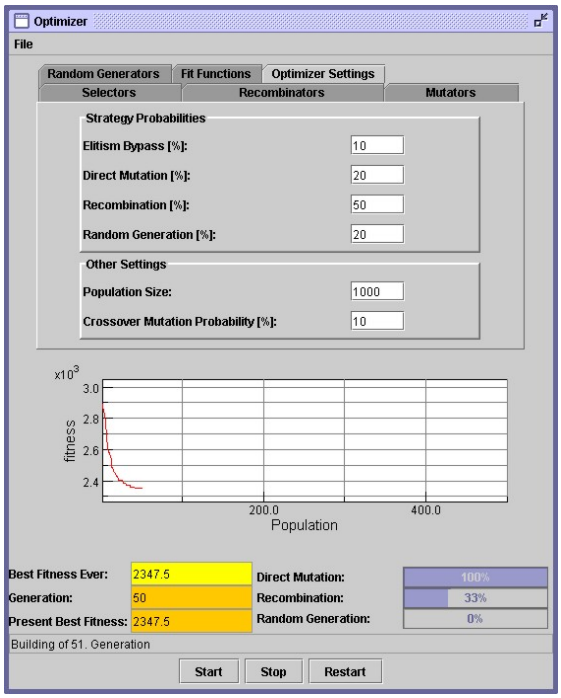

Figure 14: Screenshot of the optimizer dialog box (Optimizer Settings) of GAPO

In the optimizer dialog box the user can choose any combination of genetic operators as well as any combination of fitness functions. This gives the possibility to adjust the optimization to the own needs.

The graph in Figure 14 shows a steep decline in the chosen fitness function combination indicating that the quality of the project plans improve rather quickly during the first twenty to thirty generations. This behavior is typically for all encountered optimization runs so far. If only the *ProjectDurationFittnessFunction* is applied, the optimization reduces the project duration by about 20% against a first generated project plan. If only the *AreaFitnessFunction* it considered, the total use of resources is again optimized by about 15% to 20%. Of course, these numbers only apply to the specific problem but we expect a higher optimization potential in bigger and more complex projects. In addition, in real life the project costs are normally more stringent then the duration, although these two are correlated to some degree.

4.4 Conclusions

We have demonstrated that real life project plans can automatically be created and optimized with our modular application GAPO. In future work we have to apply our method to a number of more complex, practical projects in order to investigate the efficiency of our GA-operators and fitness functions. Most probably more appropriate operators have to be found and fitness functions specific to customer needs must be developed. Further, our optimizer requires a lot of parameters to be set by the user which is rather cumbersome and should at least partially be automated. This means that the optimizer should be able to adapt its settings according to the progress during the optimization. This is what we would consider as an evolutionary algorithm with the final goal, that the user could handle the GAPO as a black box, not requiring any knowledge about optimization, genetic algorithms and the like.

5 Endnotes

Authors:

Farbian Märki, Dipl. Informatiker FH, AEC – Informatics Institute, University of Applied Sciences Aargau, Steinackerstrasse 5, CH 5210 Windisch

Marc A. Suter, Dipl. Informatiker FH, AEC – Informatics Institute, University of Applied Sciences Aargau, Steinackerstrasse 5, CH 5210 Windisch

Prof. Dr. Manfred Vogel, AEC – Informatics Institute, University of Applied Sciences Aargau, Steinackerstrasse 5, CH 5210 Windisch

Prof. Dr. Ing. Manfred Breit, AEC – Informatics Institute, University of Applied Sciences of both Basel, Gründenstrasse 40, CH 4132 Muttenz

6 References

Aalami, F. B., Kunz, J. C. and Fischer, M. A. (1998). *Model-Based Mechanisms for Automated Activity Sequencing*, Department of Civil and Environmental Engineering, Stanford University, Stanford, CA 94305-4020, USA

Mettler, K. and Krauer; M. (1999), *Bocad-CMM-Interface.* Diploma thesis, unpublished, University of Applied Sciences Aargau, Switzerland

Raps, M. (2002), *4D Modellierungen.* Diploma thesis, unpublished, University of Applied Sciences Aargau, Switzerland