Construction Scheduling Optimization by Simulated Annealing

M. König[1] and U. Beißert[2]

[1]Chair of Theoretical Methods for Project Management, Bauhaus-University Weimar, Marienstr. 7, 99423 Weimar, Germany, PH (49) 3643-584825; FAX (49) 3643-584565; e-mail: markus.koenig@uni-weimar.de
[2]Chair of Construction Management, Bauhaus-University Weimar, Marienstr. 7, 99423 Weimar, Germany, PH (49) 3643-584455; FAX (49) 3643-584565; e-mail: ulrike.beissert@bauing.uni-weimar.de

Abstract

The process of generating optimized schedules for construction projects is a very time-consuming. For each construction task several execution restrictions, conditions and requirements such as technological dependencies or resource availabilities have to be considered. This leads to a multitude of possible execution orders and consequently to a hard optimization problem. Within this paper the integration of a flexible Simulated Annealing metaheuristic into a constraint-based simulation concept is presented to determine optimized construction schedules. Simulated Annealing approximates the optimized solution by defined temperatures and cooling rates. Both parameters help to define the search process and respectively help to escape from local minima. In the following the Simulated Annealing approach and its implementation are described in detail. Furthermore, the application of Simulated Annealing is demonstrated using an example from construction scheduling.

Introduction

Scheduling the execution processes for a construction project is a challenging task. Resources such as machines and employees, as well as spatial restrictions of the construction site, must be considered in the scheduling of construction tasks. Additionally, the site layout changes during processing, which requires material and transport flows to be adapted. All-in, the execution flow is influenced by a multitude of different process and project requirements that have to be considered in detail during the planning phase. This leads to a multitude of execution sequences. Considering the project objectives, like the minimization of the total execution time or costs, an eligible alternative has to be generated and selected.

Simulation models are successfully applied within the manufacturing industry to improve production flows. The efficiency of production facilities, local plants or specific production lines can thereby be evaluated, and material flows as well as employee utilizations can be optimized. The application of simulation models is also a promising approach for planning various processes in the construction industry. Simulation enables users, for example, to visualise material flows, to localize manpower bottlenecks or to run what-if scenarios. Due to the fact that simulation applications in the manufacturing industry only support static layouts, a dynamic and flexible simulation approach is required to describe complex construction processes.

Within the research collaboration SIMoFIT (Simulation of Outfitting Processes in Shipbuilding and Civil Engineering) a constraint-based simulation approach was developed to improve execution scheduling of civil engineering and the shipbuilding industry (König et al. 2007). This approach allows, process and project conditions as well as current as-is states to be easily integrated by defining or removing constraints. During a simulation run the fulfillment of constraints is continuously checked and therefore only valid execution schedules are generated (Beißert et al. 2007).

The Simulated Annealing approach is integrated into the constraint-based simulation concept in order to calculate optimized schedules. Constituting on an initial schedule Simulated Annealing attempts to improve neighboring schedules by task substitution. Once an improved schedule is determined, the new solution replaces the current solution. Simulated Annealing enables declined solutions to be accepted in order to escape from local minima. The procedure and its consideration within the current simulation concept are presented within this paper in detail. Finally, a case study looking at the scheduling of several finishing trades is presented to evaluate the Simulated Annealing approach.

Constraint-based Simulation

Construction scheduling problems can be described by constraint satisfaction which is a powerful paradigm for modeling complex combinatorial problems (cf. Blazewicz et al. 2007). Classical constraint satisfaction problems are defined by sets of variables, domains and constraints between the variables (Rossi et al. 2006, Kumar 1992). Accordingly, when modeling the construction scheduling problems as constraint satisfaction problems, the construction tasks, material, employees, equipment, and the construction site layout are represented by variables. Subsequently, different scheduling constraints can be specified between these variables. These constraints have to be satisfied before the associated construction task can begin. Some typical construction constraints are formalized in Beißert et al. (2007) and shown in Table 1.

Table 1: Typical constraints for construction scheduling

Construction constraints
• Technological dependencies • Capacity, e.g. of equipment, employees • Availability, e.g. of material • Spatial aspects, e.g. safety or working areas

Variables and constraints can be represented by so-called constraint graphs. Thereby, for each constraint type an associated constraint graph can be calculated. Within this paper these graphs are used for the presentation of technological dependencies. The solutions of constraint satisfaction problems are valid execution orders of the construction tasks, where all associated technological constraints are fulfilled. Normally, solving such complex constraint satisfaction problems is extremely time-consuming. The constraint-based simulation can be used to generate a possible solution very quickly (cf. Beißert et al.). Therefore, the constraint satisfaction approach was integrated into a discrete event simulation application.

During discrete event simulations different events are generated by the procedures starting tasks and stopping tasks (cf. König et al.). Thus, the simulation time leaps from event to event. Furthermore the demand for the fulfillment of constraints is controlled within the procedures. A task can only be scheduled if all associated constraints are fulfilled. In Figure 1 the procedure of starting tasks is depicted. If a new event occurs, all tasks that have not been started are checked on fulfillment of their associated constraints. This leads to a set of next executable tasks. In the next step one of these executable tasks is selected for starting. Presupposed objects of this selected task like material, resources, or employees are locked during its execution and cannot be used by other tasks. Subsequently, all not-started tasks have to be checked again on fulfillment of their constraints by going to step one. The procedure is repeated until no more tasks can be started at the current time. If the remaining time of a construction task has expired, the task is marked as finished. Its presupposed objects are unlocked and can be used by other construction tasks.

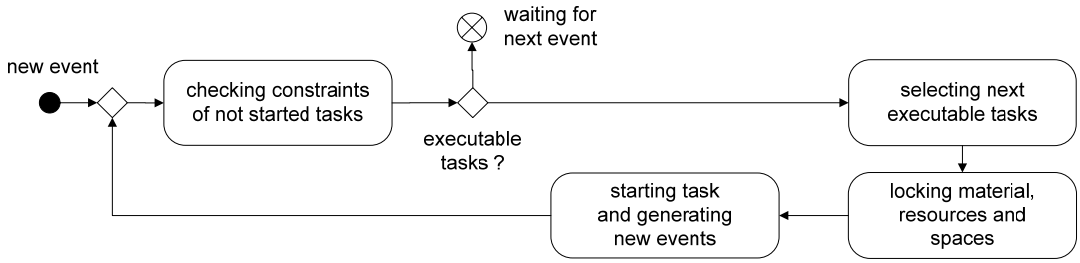

Figure 1: UML-diagram of starting tasks

The starting and stopping of construction tasks are repeated until all tasks have been completed. All events as well as the locking and unlocking of material, resources, equipment, and working spaces are recorded. Thus, each simulation run leads to a practicable construction schedule that can be investigated afterwards, for example with regard to time or costs.

Simulated Annealing

Simulated Annealing is a well-known local optimization approach for solving complex combinatorial problems like construction scheduling problems. The general goal of local optimization methods is to find good solutions in an adequate amount of time. The concept of Simulated Annealing is inspired by the physical annealing process in metallurgy (Kirkpatrick et al. 1983, Cerny 1985). In this context, physical annealing is known as the heating and controlled cooling of metal to bring the material structure from an arbitrary initial state to a state with the minimum possible energy. During heating, the metal atoms become unstuck from their current position and arrange themselves randomly. The slow cooling phase allows the atoms to find highly structured configurations with lower internal energy than in the initial configuration (cf. Aarts et al. 2005, Dréo et al. 2006).

If this physical process is considered as an analogy to our area of concern, the solutions of an optimization problem represent the possible configurations of the atoms. The objective value of a solution, the so-called cost factor, is equivalent to the internal energy state. Starting with a high temperature and a randomly selected initial solution, the Simulated Annealing heuristic calculates a new solution within the neighborhood of the current solution. The acceptance of new solutions is based on a probability that depends on the difference between the corresponding costs and on the current temperature. Ultimately this means, that high temperatures allow the acceptance of new solutions, which causes higher costs. The probability of accepting higher costs decreases within the optimization process. Once accepted, the new solution is the starting point for the next optimization step. Consequently, in order to use the Simulated Annealing heuristic an appropriate neighborhood, a good temperature-based probability and an effective decreasing rate for the temperature have to be specified.

Neighborhood

The definition of an appropriate neighborhood for a current scheduling solution is very important. The construction tasks and their associated technological constraints form a directed acyclic constraint graph. Topological ordering is used to calculate an execution rank for each task (cf. Pahl and Damrath 2000). The rank depends on the ancestor degree of the task and is determined by the maximum length of connected ancestors within the graph. Thus, tasks with the ancestor degree of k belong to the rank k. Consequently if all needed resources were available in unlimited quantities, all tasks with an identical rank could be executed simultaneously. Typically, however, construction tasks with the same rank can only be executed successively, depending on the resource capacities and the current resource allocations. This leads to a partial task order within each rank.

Within the scope of our Simulated Annealing approach the local neighborhood of a schedule is defined as the substitution of two construction tasks of the same rank. Thus, a new solution can be generated by selecting two different tasks of each rank for this substitution randomly. Figure 2 shows the topological ordering of the technological constraint graph of a simple scheduling problem. The problem consists of seven tasks {A, B, C, D, E, F, G} that have to be executed by four different resources {R1, R2, R3, R4}.

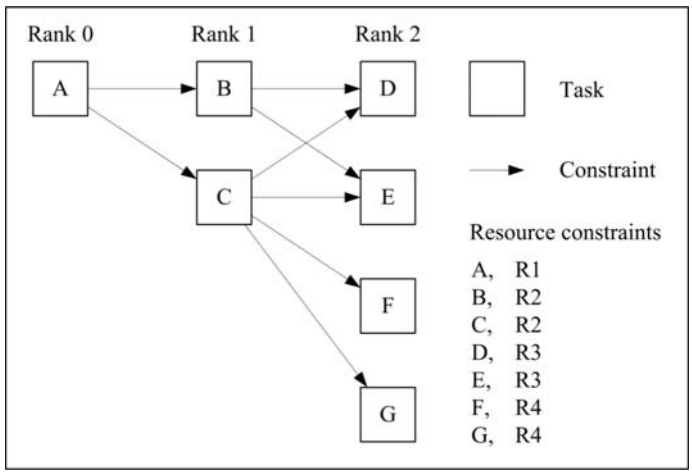

Figure 2: Topological ordering of construction tasks

The resource constraints are not considered when determining the topological ordering. Nonetheless, these constraints have a deep impact on the resulting schedules and therefore on the resulting makespan and project costs. An initial solution can be generated based on the specified resource requirements (cf. Figure 3). Within this solution each rank has also an initial execution order of its construction tasks. The initial execution order of rank one is < B, C > and of rank two is < D, F, E, G >.

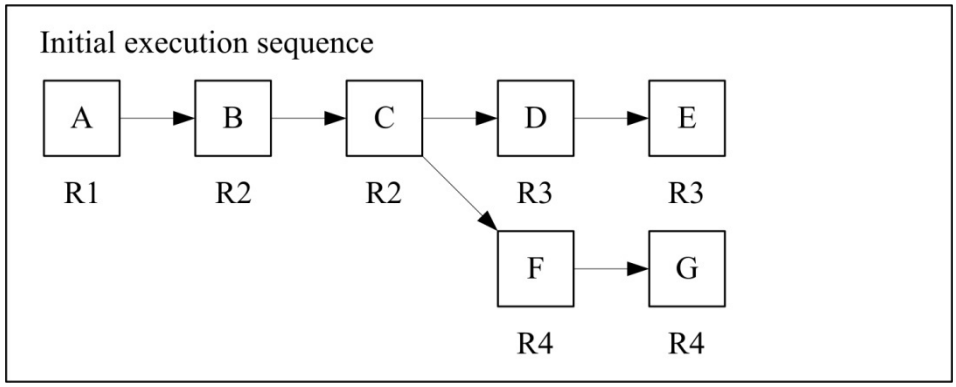

Figure 3: Initial execution sequence

Considering this initial solution a neighboring new solution can be generated by interchanging the execution positions of the tasks B and C for rank one. This new solution also fulfills the defined constraints and leads to another correct resource allocation. Now, the new partial order of rank one is {C, B} and of rank two is {E, F, D, G} (cf. Figure 4). In a second optimization step the execution positions of the tasks D and E belonging to rank two can be substituted.

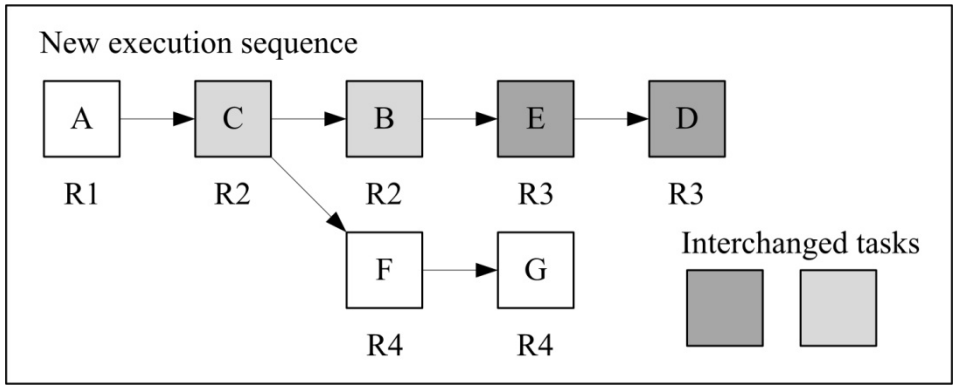

Figure 4: New neighboring execution sequence

Rules of acceptance

The probability of accepting a solution as a new start candidate can be specified by a probability function p(cc, cn, t), which depends on the current costs cc, the costs of the new solution cn and the current temperature t. Kirkpatrick et al. (1983) defined a typical probability function for many optimization problems as p(cc, cn, t) = 1 if cn < cc, and exp((cc – cn) / t) otherwise. This means that new solutions with lower costs are always accepted. When the temperature t goes to zero, the probability p tends towards zero.

The convergence speed of the Simulated Annealing approach to find good solutions depends on the initial temperature ti, the criterion for decreasing the temperature and the decreasing rate Δt of the temperature. Ideal values for these parameters cannot be determined beforehand. These control values strongly depend on the specific optimization problem and have to be empirically adjusted. Normally, the decreasing criterion is based on the length of homogeneous Markov chains or a number of maximal iterations for the same temperature (cf. Dréo et al. 2006). Different static and dynamic cooling concepts exist for decreasing the temperature (Aarts et al. 2005). Within our case study the decreasing criterion is defined by a maximum iteration number and the decreasing rate Δt of the current temperature t as $\Delta t = t – (\alpha * t)$

with the cooling parameter α. Typical values for α vary between 0.8 and 0.99 (Aarts et al. 2005). The detailed Simulated Annealing algorithm is shown in Figure 5.

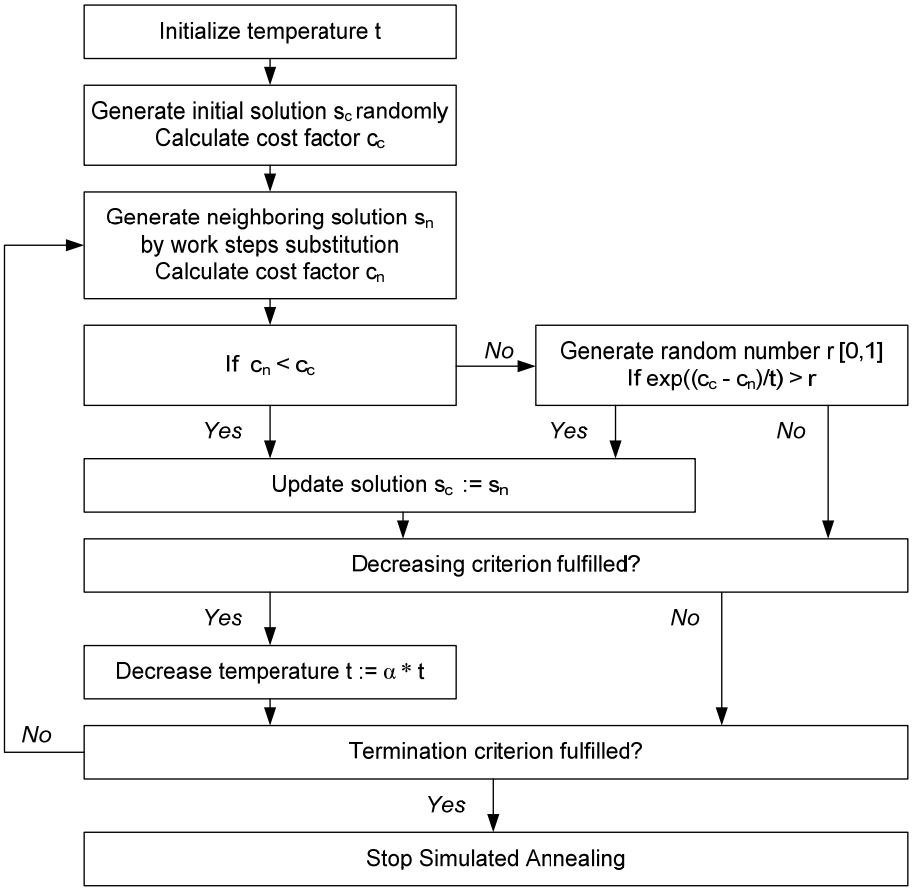

Figure 5: Simulated Annealing procedure according to Parmee (2001)

Implementation

Simulated Annealing is integrated into our constraint-based simulation approach by the following implementation steps. At the beginning of a new optimization the ranks of all construction tasks are generated, grouped, ordered, and stored in an execution list l. Furthermore the initial start temperature ti, the cooling parameter α, and maximum number of iterations for the same temperature are specified. For the generation of a new solution within a restricted neighborhood, the routine starting tasks (cf. Figure 1) was extended by the step "ordering tasks considering the restricted neighborhood" (cf. Figure 6). Therefore a temporary execution list lt is applied where the execution order of the tasks for each rank are stored. During a simulation run this list is continuously updated. After a list of next executable tasks is generated, its execution position within its associated rank is calculated for each task. Therefore, the temporary execution list is checked and the new results are added to the list. Thus, the execution sequence of each rank can be exactly specified afterwards.

Within the first simulation run the next executable tasks are started randomly. Based on the temporary execution list lt and the neighborhood definition further execution orders of the tasks are calculated for all subsequent simulation runs. New solutions are generated by random substitution of two different work steps of the same rank. The interchanging is started with the lowest ranks. If there are two or more next executable tasks listed belonging to the same rank, then two of them are interchanged. Within each optimization step only one rank modification is performed. Each generated execution order is stored in a Tabu list. If a new order already exists in the Tabu list, the ordering routine for the current event is repeated by interchanging other tasks iteratively until a new ordering is found or a termination criterion is fulfilled. The termination criterion is defined by a maximum number of iterations.

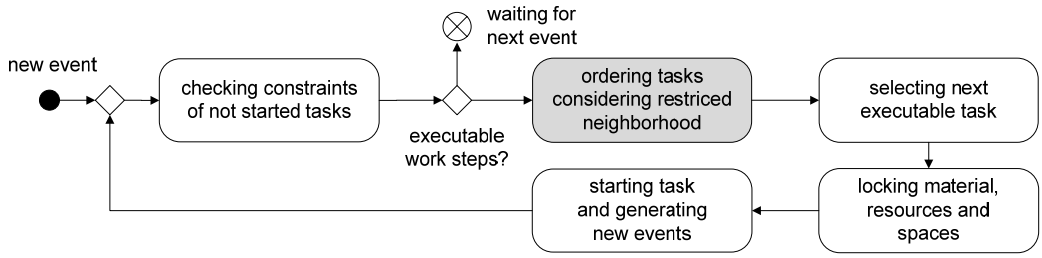

Figure 6: Generating a new solution by ordering executable tasks

After each optimization step the list lt contains a new execution order for all tasks. Consequently, the new costs cn for this new schedule can be calculated. In the next step, the probability value for accepting the new solution is specified and checked by using a normalized random number r. The new solution is accepted, if r is less or equal p(cc, cn, t), then the temporary execution list lt is copied to the list l. Following this, the current temperature t is reduced by Δt, if the maximum number of iterations for this temperature t is achieved.

Case study

In this section the presented Simulated Annealing concept is evaluated by scheduling several finishing trades of a building storey with twelve rooms, thirteen dry walls, twelve interior doors, and fourteen window sills (cf. Figure 7).

Building story (100 x 69 foot)

- 12 rooms
- 13 dry walls
- 12 interior doors
- 14 window sills

Figure 7: Finishing elements of building storey

The different finishing trades with their considered work steps are shown in Table 2. Based on the given number of rooms, dry walls, window sills, and interior doors as well as the specified finishing trades, 190 tasks and 1199 technological and resource constraints are generated.

Table 2: Finishing trades with tasks and required workers

Finishing Trades	*Tasks and required workers*
Ground works	spill floating screed (two floorer), lay flooring tile (one floorer), lay carpet (two floorer)
Dry construction	install drywall (two drywaller), install hung ceiling (two drywaller)
Joinery	install window sill (one joiner), install doors (one joiner)
Coating	color wall (one painter), color ceiling (one painter)

Figure 8 depicts the partial constraint graph for one floor, two hung ceilings, and two dry walls including the opening and coating tasks. In this example, the construction task *spill floating screed* has to be finished before covering tasks in the same floor can be started. Furthermore, before the construction tasks *spill balancing material* can be started the two associated walls have to be assembled completely.

For this case study two drywallers, two painters, two floorers as well as one joiner were specified. Under the given resource constraints (cf. table 2), optimized schedules with minimal makespan are determined by

Monte Carlo simulation and Simulated Annealing. Both optimization experiments were simulated for exact 60 minutes. Within the Monte Carlo experiment the next executable tasks are selected randomly. For the Simulated Annealing experiment an initial temperature ti = 300, a cooling parameter $\alpha = 0.8$ and the same maximal number of iterations for the ordering routine (cf. section implementation) and for the temperature reduction of n = 10 were used.

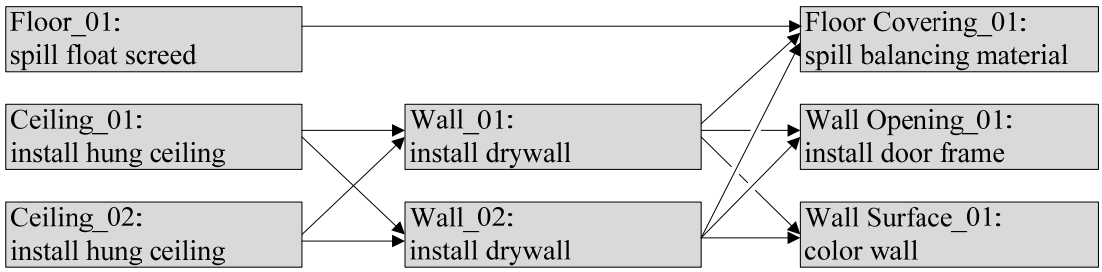

Figure 8: Partial constraint graph of selected finishing tasks

The results of the Monte Carlo and Simulated Annealing optimization are shown in Table 3. After 10 minutes 1000 different Monte Carlo schedules with an overall minimal makespan of 535 hours were simulated. By using Simulated Annealing 137 accepted solutions were generated. The last and therefore best solution has a makespan of 492 hours. In this case study the Simulated Annealing heuristic found an execution schedule that is about 43 hours faster than the best schedule generated by Monte Carlo simulation.

Table 3: Evaluation of the optimization experiments

Experiments (10 minutes)	Max [h]	Min [h]	$\bar{x}$ [h]	σ [h]
Monte Carlo	658	535	632	40.44
Simulated Annealing	627	492	525	19.63

Conclusions and Outlook

Using the constraint-based simulation approach different practical schedules can be simulated. Afterwards promising solutions can be evaluated in terms of work and material flow organization, utilization of space and worker's efficiency as well as process costs, afterwards. However, the application of Monte Carlo Simulation is very time-consuming and the locating of optimal respectively optimized schedules is not guaranteed. The application of local optimization methods guarantees the calculation of optimized schedules in an adequate amount of time. We integrate the Simulated Annealing metaheuristic in our simulation concept. Thus, clearly improved solutions can be generated in a diminished amount of simulation runs.

In future work, the justifications of the heuristic parameters like temperature and its cooling rate have to be investigated. The parameters choice is decisive for results quality. Thus, a problem-specific adaptation of both is essential to effective generate good schedules. The decreasing criterion based on the length of homogeneous Markov chains or a number of maximal iterations for the same temperature are promising possibilities for a problem-specific adaptation and are investigating in future work.

References

[1] Aarts, E., Korst, J., Michiels, W. (2005). "Simulated Annealing". In "Search Methodologies - Introductory Tutorials in Optimization and Decision Support Techniques" edited by Burk, E. K., and Kandall, G., Springer, New York

[2] Beißert, U., König, M., Bargstädt, H.-J. (2007). „Constraint-based simulation of outfitting processes in building engineering". 24th W78 Conference, Maribor, Slovenia

[3] Blazewicz, J., Ecker, K., Pesch, E., Schmidt, G., and Weglarz, J. (2007). „Handbook on scheduling: from theory to applications". Springer, Berlin

[4] Cerny, V. (1985). "A thermodynamical approach to the travelling salesman problem". Journal of Optimisation Theory and Applications, Vol.15, 41-51
[5] Dréo, J., Siarry, P., Petrowski, A., Taillard, E. (2006). "Metaheuristics for Hard Optimization: Methods and Case Studies". Springer, Berlin
[6] Freuder, E.C., Wallace, R. (1992). "Partial constraint satisfaction". Artificial Intelligence 58, 21–49
[7] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P. (1983). "Optimization by Simulated Annealing". Science, 220, 671-680
[8] König, M., Beißert, U., Steinhauer, D., Bargstädt, H.-J. (2007). "Constraint-based simulation of outfitting processes in ship building and civil engineering". 6th Eurosim Congress in Modelling and Simulation, Ljubljana, Slovenia
[9] Kumar, V. (1992). "Algorithms for constraint satisfaction problems: A survey". AI Magazine 13(1), 32-11
[10] v. Laarhoven, P.J.M., Aarts, E.H.L., Lenstra J.K. (1992). "Job shop scheduling by simulated annealing". Operations Research 40(1), 113-12
[11] Pahl, P.J., Damrath, R. (2000). "Mathematische Grundlagen der Ingenieurinformatik". Springer, Berlin
[12] Parmee, I.C. (2001). "Evolutionary and adaptive computing in engineering design". Springer, London
[13] Rossi, F., van Beek, P., Walsh, T. (2006). "Handbook of Constraint Programming". Elsevier, Amsterdam