Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, eds

DETERMINATION OF FLOAT TIME FOR INDIVIDUAL CONSTRUCTION TASKS USING CONSTRAINT-BASED DISCRETE-EVENT SIMULATION

Gergő Dori

Technische Universität München
Arcisstrasse 21.
D-80333 Munich, GERMANY

André Borrmann

Technische Universität München
Arcisstrasse 21.
D-80333 Munich, GERMANY

ABSTRACT

In this paper we propose a methodology which combines Forward and Backward Simulation in order to extend discrete-event simulation by the ability to calculate float times. In the case of Backward Simulation, the simulation starts at the virtual completion date of the construction project and runs backwards in time until the start date. To determine float times it is important that the task execution sequence is in the same order for forward and backward simulation. Consequently we present an extension to the simulation concept that controls the execution order of the tasks within the backward simulation. By combining the results of the forward and backward simulation, it is possible to determine float times for each task while taking into account resources. A comprehensive case study illustrates the application of this new approach. One result of this is the determination of detailed float times for each task using discrete-event simulation.

1 INTRODUCTION

The creation of a detailed schedule for a construction project considering the available resources and precedence relationships between construction tasks is a challenging task for construction managers. The key measure of flexibility within the schedule is the *float time* (Raz et. al. 1996). It describes the time frame within which the execution of a task can be moved, or the duration leeway of a task without impacting on the total duration or the execution of subsequent tasks. The standard approach to determining float times in today's construction-process planning is the Precedence Diagram Method (PDM), including well-established methods such as the *Critical Path Method* (CPM) or *Program Evaluation and Review Technique* (PERT). Where material and resource restrictions do not need to be considered, i.e. only precedence relationships of tasks are crucial; the schedule can be analyzed by applying CPM or PERT methods and represented by an activity-on-node diagram. In this case the calculation of float times for tasks is a straightforward procedure. Starting with the first task, the execution time will be added to the start time. The result is the *earliest end time* of the first task as well as the *earliest start time* of the successor tasks. After finishing this so-called *forward-pass*, the earliest possible start time is calculated for each task. Subsequently, a *backward-pass* is conducted, which works in a similar manner. Starting with the last tasks, i.e. those that have no successors, the *latest start and end times* of all the tasks are calculated. The difference between the latest and the earliest start times defines the *total float time* of a task. The tasks without float times constitute the *critical path*. A delay in any of these tasks will result in an increase of the project's overall time span and therefore in a delay of the project itself. For this reason it is important to determine float times for each task, so that it is possible to identify tasks that are part of the critical path or have only short float times.

If the availability of resources is limited for a project, the calculated results can be misleading. Using the CPM method alone it is not possible to consider anything other than precedence constraints and as a result CPM cannot determine feasible schedules and float values under limited resource availability. These kinds of scheduling problems, where limited amount of resources are assigned to construction tasks fall under the category of *Resource-Constrained Project Scheduling Problems* (RCPSP). In this paper we pre-

sent a concept that makes it possible to calculate float times for all tasks in a resource-constrained project schedule.

Previous research investigated the generation of valid schedules for construction scheduling problems by applying *constraint-based discrete-event simulation* (Beißert et al. 2007; König et al. 2007). This approach is able to generate construction schedules that consider not only precedence relationships but also resource and material constraints. Using this simulation technique, the earliest start time for each task can be determined. In this paper a concept is introduced that extends constraint-based discrete-event simulation by the ability to determine float time for construction tasks. In a manner similar to the forward- and backward-pass analysis of the CPM, this approach combines the forward and backward simulation concept with additional methods that apply restrictions to obtain an identical order of tasks for the backward simulation and the forward simulation.

The following section gives a review of different approaches in the available literature. The latest research progress on the topic of float calculation using CPM and the methodology of constraint-based discrete-event simulation is also presented briefly. Section 3 describes in detail the concept of calculating float times using combined forward and backward simulation. A comprehensive case study is presented in section 4, while the final section 5 contains concluding remarks and an outlook on future research objectives.

2 RELATED WORK

The most widely applied approach in today's construction-process planning is the CPM. As Kelley (1961) has shown, this approach is optimal for construction schedule planning because it minimizes the overall project duration. However, scheduling under resource constraints is a much more complex problem. Raz and Marshall (1996) point out that the results can be misleading when the CPM is applied to float determination while taking limited amount of resources into account. Using CPM alone, the situation can occur where the resource limits will be exceeded. To solve this overutilization problem, heuristic methods are applied to level the resources and decide to which of the competing tasks the resources should be allocated first so as not to exceed the resource limits (Willis 1985, Raz et. al. 1996, Hegazy 2011). As a result, some tasks may be scheduled later than their late dates (with CPM) because the required resources are not available. This conflicts with scheduled early dates and late dates respectively, and the float times are no longer applicable.

To solve this problem, Raz and Marshall (1996) propose new float definitions that retain the original meaning of float. They introduce the *scheduled total* (and *scheduled free*) *float* which is the difference between the latest scheduled date and earliest scheduled date of a task. Scheduled dates fulfil both the precedence and resource constraints of the task and the resource limits of the project. To determine these scheduled dates a resource-constrained forward- and backward-pass is carried out using heuristic algorithms for resource leveling. Using their approach it is possible to obtain a more precise and accurate measure of schedule flexibility and risk. Goldratt (1997) introduces the definition of the *critical chain*, a series of tasks satisfying both precedence and resource constraints (equals critical path with satisfied resource constraints and limits). A delay in one of these chain tasks would extend the total duration of the project. *In this paper, we will also use this definition for critical tasks taking resource constraints into consideration*. Bowers (2000) redefines the definition of float time in three types. He calculates the float time as the difference between the earliest and the latest start time of the task of *all possible schedules* with identical durations. Lu and Lam (2008) state, that CPM is not able to handle multiple resource calendars for the total float determination. They propose a new method for eliminating this drawback based on forward-pass analysis alone. They generate a schedule under resource limit and resource calendar constraints and obtain the base project duration. They then select a task and extend its duration iteratively by one day. In every iteration step a resource leveling analysis is run and the project duration is updated. As long as the updated project duration is not greater than the base project duration the total float time of a task gains is extended by one extra day. This process continues until all the float times for all the tasks have been calculated.

Moreover, Hegazy and Menesi (2010) point out that the CPM algorithm has no mechanism for considering multiple resource constraints, such as deadline and limited resources. It is therefore often difficult to generate schedules that are feasible for all constraints, i.e. one solution in response to one constraint may conflict with another solution for another constraint. Hegazy and Menesi's proposed solution is a mechanism that is based on a finer level of granularity. This is achieved by breaking down the task duration into separate time segments. As described above, researchers have proposed improvements to CPM, for example to obey more and correct information about float times. In addition to the segmentation technique, the researchers have proposed alternative heuristic approaches e.g. genetic algorithms to generate a schedule that can satisfy both deadline and resource constraints (Hegazy and Menesi 2011).

Lim et. al. (2011) discuss the limitations of the currently used approaches in interpreting flexibility of resource-constrained projects. They point out that an appropriate definition of float cannot be based on one single schedule because multiple schedules can exist with the same project duration. Therefore they redefine the notion of critical activity with a broader scope: "*An activity is a critical activity if the maximum of its float values is zero for a specified completion time of the project*."

One other promising and accepted approach to solving RCPSP is the use of discrete-event simulation models. Past research has focused on applying and improving such models and efforts have been undertaken by Halpin (1977), Chang (1986), Tommelein et al. (1994), and AbouRizk (2011). Based on several different modelling concepts, processes, sequences and resources are described and simulated using sophisticated simulation tools. An innovative approach was also presented by Beißert et al. (2007) and König et al. (2007). Here, the constraint satisfaction approach was adapted for integration within the discrete-event simulation. The scheduling problem is described in such a way that construction tasks are modelled as variables. In addition, every restriction of a task, such as precedence relationship, material or resource constraint, is modelled as a constraint of the variables. The discrete-event simulation is able to find a value combination that fulfils the constraints for all variables, and so to generate a feasible schedule for the RCPSP. In Lu (2003) the methodology of discrete-event simulation is simplified for use in construction scheduling. As a result of this simplification, the use of construction simulation systems became an easily controllable tool. In Lu et al. (2008) a heuristic solution is presented for resource-constrained CPM. The proposed simplified simulation-based scheduling system enables the automatic formulation of resource-constraint scheduling with the shortest total project duration using particle swarm optimization. To calculate float time for the tasks in the generated project they use the aforementioned method, where the total float time is represented as the maximum amount of time that the duration of the task can increase without extending the total duration of the project.

The lack of a flexible constraint satisfaction modelling approach and the clear separation between simulation and optimization are, however, two reasons why the constraint-based discrete-event simulation approach is favoured in this paper. In addition, in our approach we wanted to calculate float time in a single calculation step without using heuristic or iterative algorithms, and the discrete-event simulation is capable of determining the expected results this way.

3 CONCEPT

The goal of our research is to calculate the *total float time for individual tasks* and to determine the *critical chain* of tasks considering *resource constraints* and *limited resource availability* in a construction project schedule generated using *constraint-based discrete-event simulation*.

In order to calculate float times, similar to the backward-pass analysis used in CPM, a newly developed backward simulation method along with a coupling process extending the common forward discrete-event simulation is introduced. Using this common constraint-based discrete-event simulation method (forward simulation) the earliest schedule date of all tasks can be determined. Accordingly, inverting the direction of the simulation will result in the latest schedule date for all the tasks considering resource availabilities. So, the difference between the resulting schedule date of the backward and forward simulation for a task represents its total float time with considering resource availability. When the float time of a task is zero, it is a part of the critical chain. Here, it is important to note that if more than one executable

task is available at a certain point in the simulation (all their constraints are fulfilled), the forward simulation selects the next schedulable task randomly. So, by applying the Monte-Carlo approach a multitude of feasible schedules can be generated (Wu et. al. 2010). However, to be able to determine the float time of a task, we need the order of the task execution sequence for the forward and backward simulations to be identical. To achieve this, the backward simulation instead employs a priority based selection to determine the next executable task rather than the random method. The priorities will be applied according to the finish date of a task in the schedule generated by the forward simulation. Using this approach the order of the executed tasks is identical for the forward and backward simulation but in reverse.

The principle of float time calculation is presented in Figure 1. Four tasks are to be simulated: A, B, C and D. B and C must follow A, and C must follow B and C. The execution time for each task is different. When simulating these tasks using forward simulation, B and C will start right after A has been finished, and D will start after B has been finished (C finished earlier). If we simulate these same tasks backwards, B and C will performed right after D has been completed and A will be performed after B has been completed (C finished earlier) due to the reversed direction of the simulation. The two results can be combined with one another because the complete execution time and the order of the tasks are the same. Task C has a float time between the finish of task A and the start of task D. The critical path of the schedule is therefore A→B→D because they have zero float time.

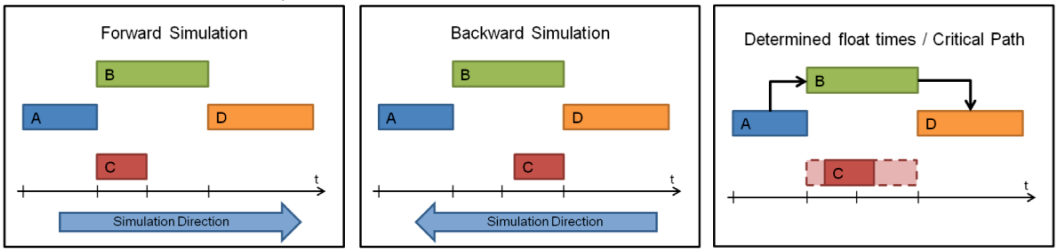


Figure 1: Concept of determining float time and critical path by combining forward (left) and backward simulation (centre).

In order to take resource limitations into account not only for the scheduling process but also for float time calculation, the resource limitations will also be considered during float time calculation. Therefore, if the resource limits within a float time range of several tasks will be exceeded, the float time of a task with lower priority will be shortened. In the example shown in Figure 2, task D has a higher priority at flow time calculation than task C, so the float time of C will be shortened by the execution length of task D. We use the principle that if a task ends later in the schedule, it will have a higher priority in float time determination then other tasks that end earlier. In this way we assume that the relative execution order of the tasks will not change during the execution and if the first task is delayed, its successors must also be delayed because of the limited resources.

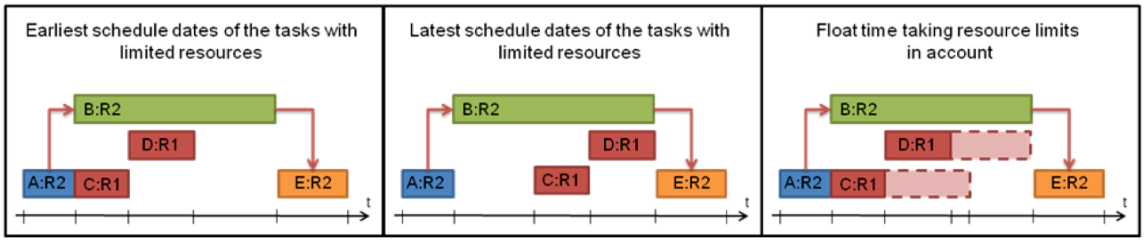


Figure 2: Determination of float time taking resource limitations into account
(tasks: A, B, C, D, E; resources: R1, R2; precedence relationships: A→B, A→C, A→D, B→E, C→E, D→E; float time: dashed boxes; critical chain: arrows)

To implement this concept and obtain reliable results from the connected simulations, conceptual changes and restrictions have to be made in the existing simulation model.

3.1 Forward Simulation

The most prevalent approach to generating construction schedules is *discrete-event simulation* (AbouRizk 2010). The *discrete-event simulation concept* was developed for mechanical engineering tasks but its advantages were quickly recognised and it was adapted for use in the civil engineering industry to simulate construction site projects. Since the introduction of the first simulation language CYCLONE (Halpin 1977), simulation models have been developed for typical construction systems providing engineers with a means to experiment with ways of achieving more productive, efficient and economical field operations (Lu 2003). Conventionally, in discrete-event simulation the execution order of the processes are predefined. This leads to a rigid and inflexible simulation behaviour, which contrasts with the more dynamic behaviour observed in real construction projects. The *constraint-based methodology* (Beißert et al. 2007) was developed with the aim of introducing more dynamism into the selection event of task execution. In this approach the construction scheduling problem is modelled as a *constraint satisfaction problem*. Here, the tasks are modelled as variables of the simulation and their requirements (precedence relationships, resource needs) as constraints. A list is created that contains all the executable tasks. The constraints of the tasks are checked and the tasks are then executed as soon as all their constraints are satisfied. The checking process stops when the list is empty and there are no more executable tasks available. The order of the tasks in this list has a big influence on the execution order of the tasks of the complete project. As a consequence, different steering methods have been introduced, e.g. random selection or ranking tasks (Beißert et. al. 2008), to implement different simulation strategies for construction projects (e.g. as fast as possible, as cost-effective as possible).

In the simulation concept presented in this paper a list of tasks is imported (Dori et al. 2011) into DESMO-J, a discrete-event based simulation engine (Page and Kreutzer 2005) that has been extended with the constraint-based methodology, resulting in the generation of a construction schedule of these tasks. To determine the execution order of the tasks a First In First Out (FIFO) list is used. A task is only put into this list when all its precedence constraints are fulfilled so that for the items in the FIFO list, only the resource constraints need to be checked.

The scheduling process begins by resource checking all tasks without predecessors; these are then transferred to the FIFO list (executable list) at the beginning of the simulation. As soon as one of these tasks is completed, its successors will be informed (because one of the precedence constraints has been fulfilled). If the successor has any other predecessors, it will wait; if not, it will be transferred to the FIFO list for scheduling as soon as its required resources are available. When multiple equivalent tasks can be started at the same point in the simulation, the selection event either carries out tasks in the order that they appear in the *executable list*, or according to a pre-defined *strategy*. Strategies are implemented by using *soft-constraints* (Beißert et al. 2008) which can define *priorities* for certain executable tasks. Using priorities, the constraint checking event will always start with the highest priority task, followed by the next highest priority task, and so on. If some or all the tasks have the same priority, the execution order will be determined by the order they appear in the executable list. By introducing this scheduling method, the earliest possible start date of a task can be calculated taking into consideration the availability of required resources.

3.2 Backward Simulation

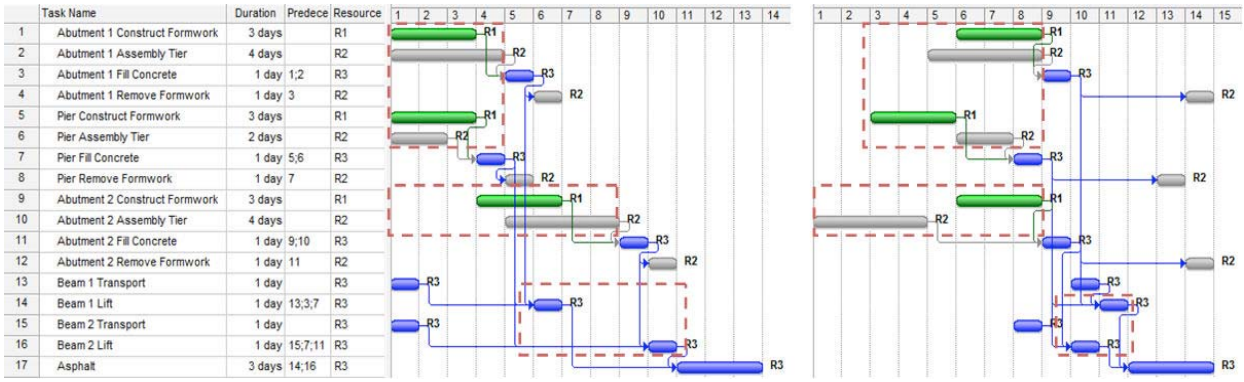
The concept of backward simulation is basically the same as that of forward simulation – the simulation actually runs forward in time but with *reversed execution conditions*. Using this approach, the resulting schedule is calculated based on a simulation that starts from a virtual completion date for the construction project and runs backwards in time until the starting point of the construction process is reached. The reversed conditions mean an inversion of the precedence constraints, e.g. for forward simulation concrete is filled after formwork assembly; for backward simulation formwork assembly follows filling with concrete (Figure 3). This entails reordering the priorities of the tasks and reversing the resource calendar. To

realize this inversion a function is written which reverses the order of the precedence constraint at the beginning of the simulation, changing the successor into predecessor and predecessor into successor.

Figure 3: Precedence relationships for forward simulation (left) and for backward simulation (right).

To test the applicability of the backward simulation approach a simple test case was utilized with unlimited resources. This configuration meant that the simulations are independent of resource constraints and the total duration of the project depends only on the precedence constraints between the tasks. The calculated results must be identical to the results determined using the CPM. Therefore the order of the executed tasks of the backward and forward simulation should be identical and the difference between the scheduled date of a task in the forward and the backward simulation is its float time. The simulations essentially reproduce the results of the CPM with unlimited resources.

By solely restricting the amount of available resources and letting the backward simulation run with the same configurations as the forward simulation causes the following issue to occur. The order of the executed tasks is not identical to the result of the forward simulation so the float time and the critical chain of tasks cannot be determined. This problem occurs because the task execution process of the simulation is based on the order of the task list, and this is not the same in the case of forward and backward simulation. To illustrate the issue, an example is presented with the construction of two abutments, a pier and two superstructure beam elements. The limited resources allow the parallel execution of only two tasks from the same resource category. The different results can be observed in Figure 4.



	Task Name	Duration	Predece	Resource
1	Abutment 1 Construct Formwork	3 days		R1
2	Abutment 1 Assembly Tier	4 days		R2
3	Abutment 1 Fill Concrete	1 day	1;2	R3
4	Abutment 1 Remove Formwork	1 day	3	R2
5	Pier Construct Formwork	3 days		R1
6	Pier Assembly Tier	2 days		R2
7	Pier Fill Concrete	1 day	5;6	R3
8	Pier Remove Formwork	1 day	7	R2
9	Abutment 2 Construct Formwork	3 days		R1
10	Abutment 2 Assembly Tier	4 days		R2
11	Abutment 2 Fill Concrete	1 day	9;10	R3
12	Abutment 2 Remove Formwork	1 day	11	R2
13	Beam 1 Transport	1 day		R3
14	Beam 1 Lift	1 day	13;3;7	R3
15	Beam 2 Transport	1 day		R3
16	Beam 2 Lift	1 day	15;7;11	R3
17	Asphalt	3 days	14;16	R3

Figure 4: Different results of forward (left) and backward simulation (right)
(colors: resources, arrows: precedence constraints, hatched boxes: changed order/earlier start).

In the results of the forward simulation, the abutment 1 and the pier is started first with abutment 2 following later. In the backward simulation, abutment 2 is started first and only later the pier and abutment 1. Due to this change in order, the lifting of the two beam elements is also swapped. Because of the changed order of the tasks, where some tasks start in the backward simulation even earlier than in the forward simulation (e.g. abutment 2 assembly tier), the results cannot be compared with each other and the float times cannot be determined.

In order to create an identical order of executed tasks for the backward simulation and the forward simulation, the two simulations have to communicate with each other, which means that the underlying simulation concept has to be modified.

3.3 Calculation of Float Times

To calculate float times with discrete-event simulation the concept of the simulations needs to be modified. The goal is to obtain an *identical order of tasks for the backwards simulation and the forward simulation*. During the forward simulation, therefore, each of the executed tasks is assigned a priority according to the finish date of the task's execution (the later it is, the higher the priority). The backward simulation then uses these priorities to replace the FIFO list with a priority-ordered list of executable tasks for every time interval, and then tries to collect the necessary resources for the one with the highest priority. If that is not possible the next task with the second highest priority is chosen and so on.

In special cases, when more tasks use the same resources and are independent from each other (no precedence constraints between them) but could be scheduled at about the same time, an issue can occur with the backward simulation. As presented in Figure 5, in the forward simulation A,B,D,F and E build a continuous chain, C is executed between A and E on a "second chain of tasks" parallel to B, D and G. C and D use the same resources and in the forward simulation C is executed before D. If the backward simulation is started, although C was executed before D in the forward simulation it can be started right after E, because D still has a successor (F) which has to be executed first. So the execution order of C and D has swapped in the backward simulation. This means, that during float time determination the resource limits will be exceeded (Figure 5).

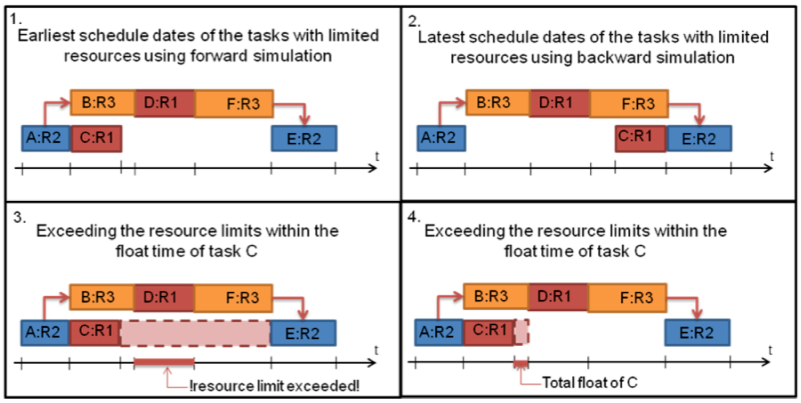


Figure 5: results of the forward (1) and backward (2) simulations
the problem (3): exceeding the resource limits at float time
the solution (4): maintain the relative order of C and D → shorter total float time for C

A further extension is therefore necessary to prevent this task order swap in the backward simulation, and to be able to calculate the total float of the tasks. An agent-based solution for a similar problem is presented by Horenburg et. al. (2012). Another favoured solution could be to apply *extra precedence constraints* between the tasks using the *same resources* based on the result of the forward simulation. This means that the order of the task execution chains for the different resource types will be kept the same for the backward simulation as it was for the forward simulation. If the amount of resources available permit more tasks to be executed in parallel, the problem becomes more challenging, and the following rules have to be considered at the generation of the new precedence constraints:

- Create precedence constraints for all the tasks in the schedule between the examined task and the successors of the *n*-th task following it in time and using the same resources; *n* is the number of tasks after the examined task where the sum of their and the examined task's resource needs first exceed (or equals) the resource limits. For example, task A (resource needs: 2) is followed by B, C and D with resource needs 2, 4, and 1 respectively; the available amount of resources is 7; starting with the first follower process B, the necessary resources are 2 + 2 = 4. 4 < 7 so the next task will be examined (C). 2+2+4 =8, which is greater than 7, so *n* = 2 and A will be constrained to the successors of C. This step makes it possible to push forward the examined task in time until the resources it uses are needed again.

- The constraints should be connected to the successors and not to the task in order to permit two tasks to be scheduled in parallel (but not later) if the necessary resources are available.
- Do not create precedence constraint for the last *m* task using the same resource type in the schedule, where *m* is the amount of parallel executable tasks (*m* depends on the available resources and the resource needs of the tasks). This step allows the free movement of the last tasks of the schedule in time, so that their float can be also calculated. Otherwise it can happen that they will be pushed back in time although the resources are available.
- The new precedence constraints will be applied after the forward simulation depending on the resulting schedule and used only for the backward simulation.

An example of the application of these extra precedence constraints is presented in the next section.

Using the methods introduced here, the execution chain of the tasks will be identical to that of the forward simulation: the backward simulation follows the task order of the forward simulated schedule, so all the tasks start later or at the same time as they do in the forward simulation. As mentioned before, the forward simulation calculates the earliest possible execution date of the tasks, so reversing the execution conditions of the simulation (e.g. the technological dependencies) will result in scheduling the tasks as late as possible in time. In order to calculate float times for each individual activity the results of the forward and backward simulation are exported to a database and compared with one another. If a task in the backward simulation starts later than in the forward simulation, the task has a float time, i.e. the difference between the two results. If the results are the same, the task has no float time and is therefore part of the critical task chain.

4 CASE STUDY

To present the application of this new approach, we will examine it using a bridge construction project. The bridge is made of two abutments, a pier and two pre-cast girder elements. The construction tasks for the abutments and the pier after constructing the foundation and for the pier ahead of the foundation, are constructing the formwork and assembling the rebars, filling the concrete and removing the formwork (for simplicity's sake we have omitted the drying time for the concrete). Before lifting the girders the corresponding abutment and the pier have to be filled with concrete, the racks for the bearings must also be filled with concrete and a support system has to be built up next to the pier. After the girders are lifted, the deck and the girder heads can be filled with concrete and the parapets can be also created. When they are ready, a sealing system can be built for the bridge, and in the end the support system for the pier can be removed.

There are three categories of resources used:

- Formwork-works and others: "casing"
- Assembly-works: "armouring"
- Concrete filling-work: "concreting"

The resources are calibrated in such a way that two tasks from the same resource type can always be executed at a time.

After exporting the results of the two simulations to a database and comparing the two results, the results of the backward simulation depending on those of the forward simulation. The results calculated only using the priority-based approach are presented in Figure 6.

These results demonstrate that using only the priority-based backward simulation, some tasks which are independent of each other can still be swapped during the backward simulation, e.g. the concrete filling task of the bearings with the concrete filling task of the pier, resulting (in this case) in a shorter schedule, but also in exceeding the resource limits during flow times. This is why the tasks of the two abutments have a large float time, and the construction of the pier becomes critical. Although all the tasks start later in time than in the forward simulation the created schedule interprets a version of the latest possible schedule dates for the tasks with predefined priority values and resource configuration rather than the real total float times because the resource limit is exceeded during the flow times. This plan could be useful if

the construction of the abutments has to be delayed for more days and changes have to be made to the schedule. In order to prevent independent tasks swapping order during the backward simulation we apply the new precedence constraints to the schedule produced by the forward simulation. The new results of the forward and backward simulation are presented in Figure 7. The order of the tasks of the backward simulation are exactly the same as of the forward simulation. So the difference of the scheduled dates of the backward simulation (latest date) and the forward simulation (earliest date) is the total float of every individual task that does not exceed the resource limits. Although delaying a task in the schedule will mean that the successors of the task will also be delayed, as long as the delay is within the interval of the presented results the total duration of the project will not be extended, and the resource limits will also not be exceeded. Consequently, the consideration of resources in the calculation of float time for construction schedules is important as it can impact on (i.e. reduce) the actual float time of individual tasks. This could not have been identified without taking resources into account (e.g. Parapet 1: fill concrete could have 3 more float days, but in that case it would collide with the girder head filling concrete task).

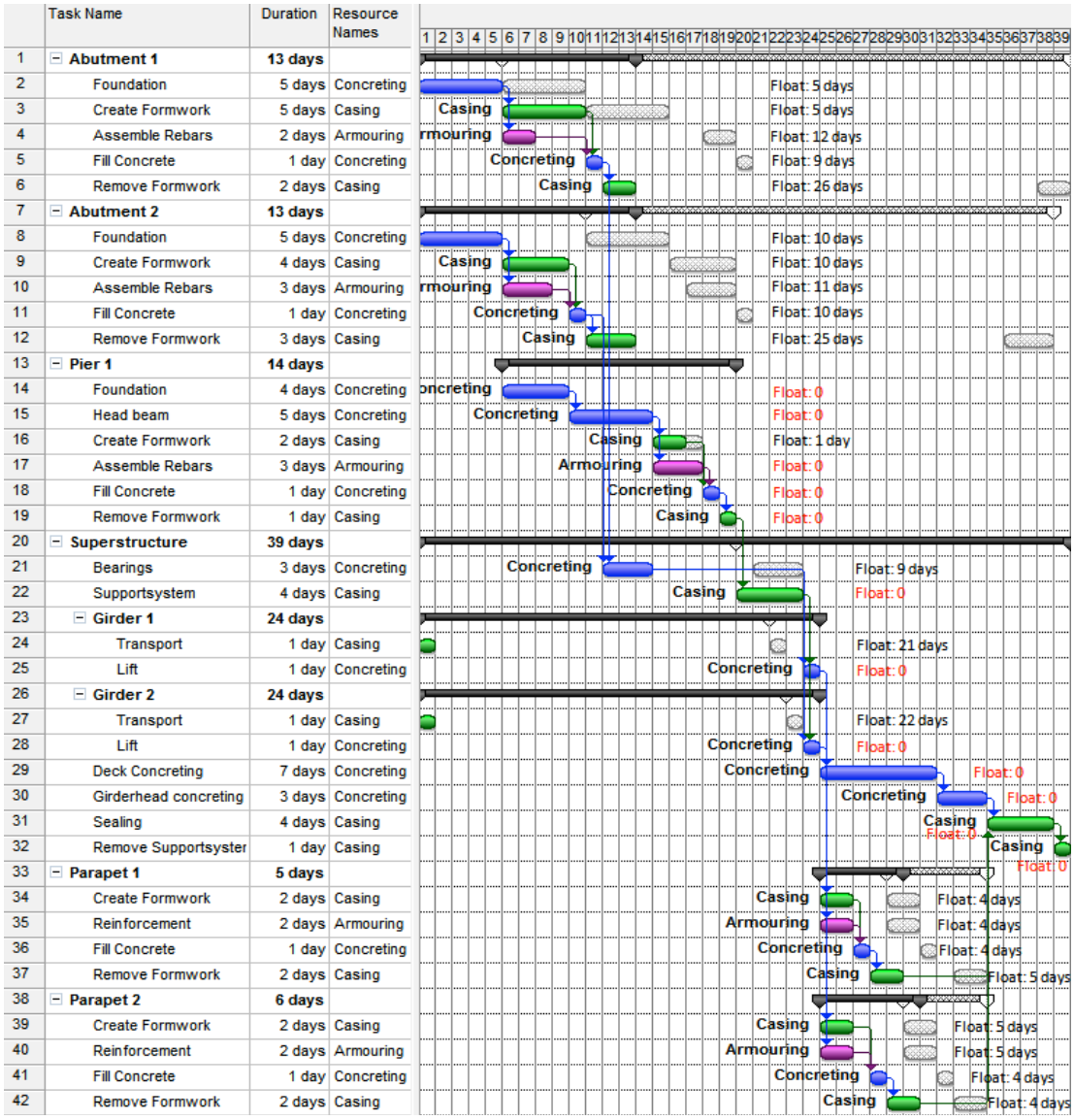
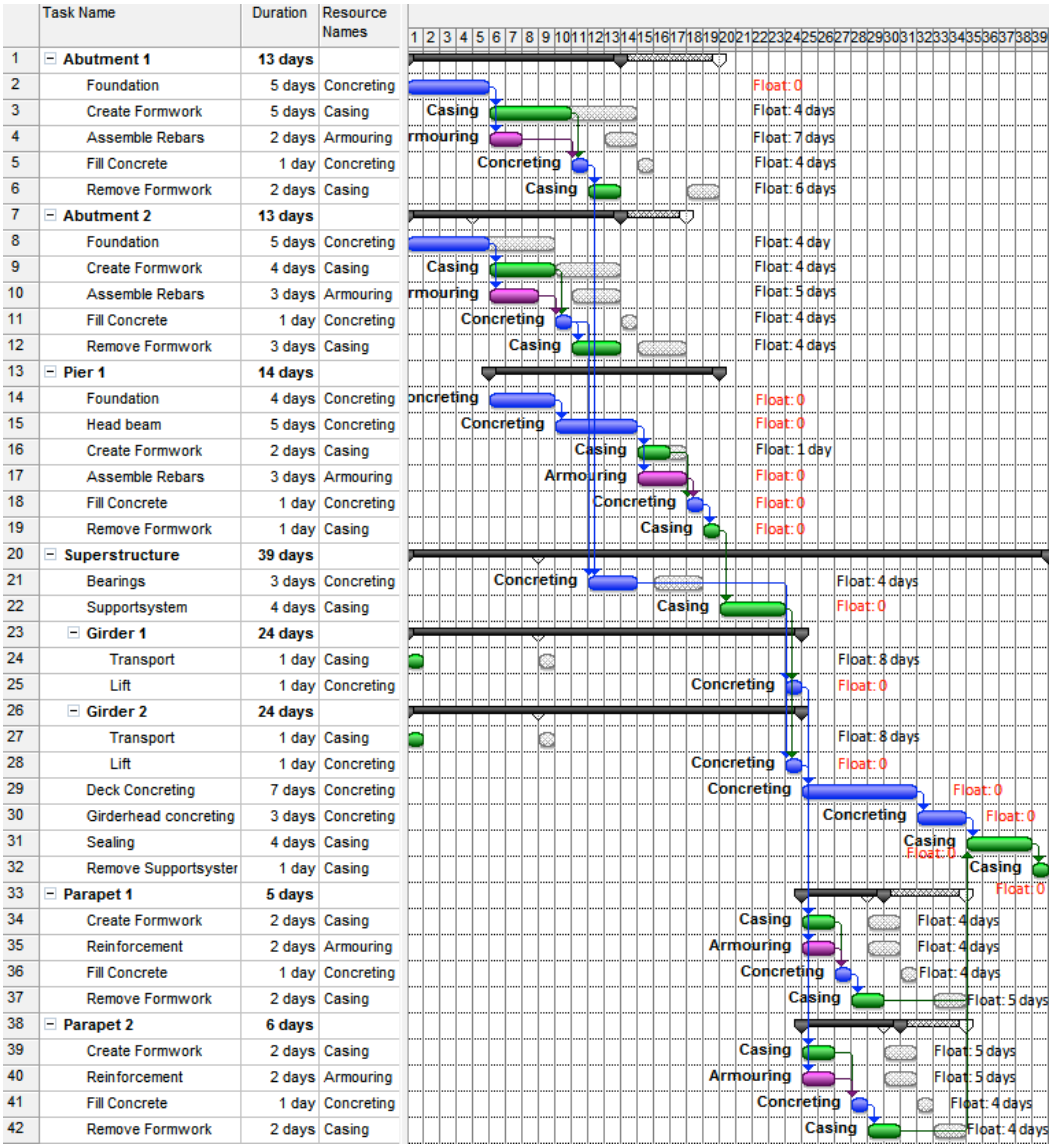


Figure 6: Results of the forward and backward simulation using only task priority based coupling (colors: resources; gray hatched areas: product of the backward simulation, arrows: precedence relationships)



	Task Name	Duration	Resource Names
1	**Abutment 1**	**13 days**	
2	Foundation	5 days	Concreting
3	Create Formwork	5 days	Casing
4	Assemble Rebars	2 days	Armouring
5	Fill Concrete	1 day	Concreting
6	Remove Formwork	2 days	Casing
7	**Abutment 2**	**13 days**	
8	Foundation	5 days	Concreting
9	Create Formwork	4 days	Casing
10	Assemble Rebars	3 days	Armouring
11	Fill Concrete	1 day	Concreting
12	Remove Formwork	3 days	Casing
13	**Pier 1**	**14 days**	
14	Foundation	4 days	Concreting
15	Head beam	5 days	Concreting
16	Create Formwork	2 days	Casing
17	Assemble Rebars	3 days	Armouring
18	Fill Concrete	1 day	Concreting
19	Remove Formwork	1 day	Casing
20	**Superstructure**	**39 days**	
21	Bearings	3 days	Concreting
22	Supportsystem	4 days	Casing
23	**Girder 1**	**24 days**	
24	Transport	1 day	Casing
25	Lift	1 day	Concreting
26	**Girder 2**	**24 days**	
27	Transport	1 day	Casing
28	Lift	1 day	Concreting
29	Deck Concreting	7 days	Concreting
30	Girderhead concreting	3 days	Concreting
31	Sealing	4 days	Casing
32	Remove Supportsyster	1 day	Casing
33	**Parapet 1**	**5 days**	
34	Create Formwork	2 days	Casing
35	Reinforcement	2 days	Armouring
36	Fill Concrete	1 day	Concreting
37	Remove Formwork	2 days	Casing
38	**Parapet 2**	**6 days**	
39	Create Formwork	2 days	Casing
40	Reinforcement	2 days	Armouring
41	Fill Concrete	1 day	Concreting
42	Remove Formwork	2 days	Casing

Figure 7: Results of the forward and backward simulation using task priority based coupling and extra precedence relationships between tasks using the same resources: total float time

When comparing the two results, we can see that as a result of applying the new constraints, the float time of some tasks with no successor is radically reduced (e.g. abutment 1: remove formwork: 26 → 6 days, or the transport of the pre-cast girder 2: 22 → 8 days). This is a product of the new constraints which for each single task restrict and define between which other tasks it must be executed. In the second case, the concrete filling task of the bearings and the pier were executed parallel with the backward simulation (these were the cause of the earlier changed execution order), so the complete order of all the tasks is the same as with the forward simulation. This is also why the construction tasks of the two abutments have shorter float times than in the first case. The critical chain of the tasks is built up as follows (indicated with red text in the figure): Abutment 1: foundation → Pier1: foundation → Pier1: head beam → Pier 1: assemble rebars → Pier1: fill concrete → Pier1: remove formwork → Superstructure: support system → Girder1: lift & Girder 2: lift → Deck "concreting" → Girder head "concreting" → Sealing → Remove support system. The total duration of the project with the given resource configuration is 39 days. The calculated total float time for the individual tasks are indicated in Figure 7. Changing the configuration of the available resources will always result in a new schedule for both the forward and

backward simulations, so all the tasks will have new float times and a new critical chain of tasks will also need to be determined.

5 CONCLUSION

Calculating float time for construction projects taking into account the available resources is a challenging task. Conventional network planning techniques (e.g. critical path method) are able to analyse and calculate float time but are not adequate for considering resources. By contrast, discrete-event simulation is capable of considering resources, but unable to calculate float time. In our research work we use a discrete-event simulation engine extended with a constraint-based discrete-event simulation methodology to generate feasible construction schedules. In this paper, we introduced a new methodology that extends the constraint-based discrete-event simulation with the ability to calculate the total float time for each individual task of the construction project. For this we used the methodology of backward simulation in combination with forward simulation. The most challenging task in the backward simulation is to follow an identical schedule sequence to that of the forward simulation. In order to achieve schedule compatibility, every task has to start at the same time or later in time than in the forward simulation. To this end, we modified the task selection algorithm so that the backward simulation uses a priority-based approach to determine the next executable task. To solve the problem where several independent tasks that use the same resources swap position in the backward simulation, new constraints are defined between these tasks based on the result of the forward simulation. This means that the latest execution date of a task will be restricted by the end date of the next task that uses the same resources, and as a result these tasks can no longer swap position.

By comparing the results with the schedule of the forward simulation, the time difference between the earliest and latest start time of a task represents its total float time. The tasks without float time comprise the critical chain of tasks for the respective configuration of resources. Finally, we used a case study to illustrate the application of this new approach and showed that the determination of detailed total float time for each individual task using discrete-event simulation taking into consideration available resources is realizable. In future work we will implement an algorithm that searches for time intervals within the schedule where the tasks with shortened float time could be also executed (within the time limit of their successors) without exceeding the resource limits, but this cannot be identified as float time.

REFERENCES

AbouRizk, S., 2010. Role of Simulation in Construction Engineering and Management. Journal of Construction Engineering and Management, 136(10), 1140-1153.

AbouRizk, S., Halpin, D., Mohamed, Y. and Hermann, U., 2011, Research in Modeling and Simulation for Improving Construction Engineering Operations, Journal of Construction Engineering and Management. 137(10), 843

Beissert, U., Koenig, M. and Bargstaedt, H.-J., 2007. Constraint-Based Simulation of Outfitting Processes in Building Engineering. In: 24th W78 Conference, 2007, Maribor, Slovenia.

Beissert, U., Koenig, M. and Bargstaedt, H.-J., 2008. Execution Strategy Investigation Using Soft Constraint-Based Simulation. In: IABSE Conference on Information and Communication Technology, 2008, Helsinki, Finland.

Bowers, J. A., 2000, Interpreting Float in Resource Constrained Projects. International Journal of Project Management 18,385-92.

Chang, D. Y.-M., 1986. RESQUE: A Resource Based Simulation System for Construction Process Planning. PhD dissertation, University of Michigan, Ann Arbor, Michigan, USA.

Dori, G. and Borrmann, A., 2011. Automatic Generation of Complex Bridge Construction Animation Sections by Coupling Constraint-Based Discrete-Event Simulation with Game Engines. In: ConVR2011, 2011, Weimar, Germany.

Goldratt, E.M., 1997, The Critical Chain, The North River Press, Great Barrington, MA.

Halpin, D. W., 1977. CYCLONE - Method for Modeling Job Site Processes. Journal of the Construction Division, 103(3), 489-499.

Hegazy, T. and Menesy, W., 2010. Critical Path Segments Scheduling Technique. Journal of Construction Engineering and Management , 136 (10), 1078-1085.

Hegazy, T. and Menesy, W., 2011. Heuristic Method for Satisfying Both Deadlines and Resource Constraints, Journal of Construction Engineering and Management, accepted manuscript.

Horenburg, T., Wimmer, J. and Günthner, W. A., 2012, Resource Allocation in Construction Scheduling Based on Multi-Agent Negotiation. In: ICCCBE 2012, Moskau, Russia.

Koenig, M., Beissert, U., Steinhauer, D. and Bargstaedt, H.-J., 2007. Constraint-based Simulation of Outfitting Processes in Shipbuilding and Civil Engineering. In: 6th Eurosim Congress in Modeling and Simulation, 2007, Ljubljana, Slovenia.

Lim, A:, Ma, H., Rodrigues, B., Tan, S. T., and Xiao, F., 2011, New Concepts for Activity Float in Resource-Constrained Project Management, Journal of Computers & Operations Research 38, 917-930.

Lu, M., 2003. Simplified Discrete-Event Simulation Approach for Construction Simulation. Journal of Construction Engineering and Management , 129(5), 537-546.

Lu, M. and Lam, H.-C., 2008. Critical Path Scheduling under Resource Calendar Constraints. Journal of Construction Engineering and Management , 134(1), 25-31.

Lu, M., Lam, H.-C. and Dai, F., 2008. Resource-Constrained Critical Path Analysis Based on Discrete Event Simulation and Particle Swarm Optimization. Automation in Construction, 17(6), 670-681.

Page, B. and Kreutzer W., 2005. The Java Simulation Book – Simulating Discrete Event Systems with UML and Java. Aachen: Shaker Verlag.

Kelley, J. E.,1961, Critical-Path Planning and Scheduling: Mathematical Basis. Operations Research 9,296-320. ONE at 1959

Willis, R.J.,1985, Critical Path Analysis and Resource Constrained Project Scheduling — Theory and Practice, Europian Journal of Operational Research 21, 149-155.

Raz, T. and Marshall, B., 1996. Effect of Resource Constraints on Float Calculations in Project Networks. International Journal of Project Management, 14(4), 241-248.

Tommelein, I. D., Carr, R. I. and Odeh, A. M., 1994. Assembly of Simulation Networks Using Designs, Plans, and Methods. Journal of Construction Engineering and Management, 120(4), 796-815.

Wu, I.C., Borrmann, A., Beissert, U., König, M., Rank, E., 2010. Bridge Construction Schedule Generation with Pattern-Based Construction Methods and Constraint-Based Simulation. Advance Engineering Informatics, 24(4), 379-388.

AUTHOR BIOGRAPHIES

GERGŐ DORI studied civil engineering at the Budapest University of Technology and Economics. After receiving his diploma in 2010 he started his Ph.D. at the Chair for Computation in Engineering at the Technische Universität München in Germany. In 2011 he joined the new Chair of Computational Modeling and Simulation, headed by his supervisor Prof. Dr-Ing. André Borrmann. His research interests are in construction site simulations and visualization. His email address is: dori@bv.tum.de.

ANDRÉ BORRMANN received his master's degree in civil engineering from the Bauhaus University Weimar in 2003 and his PhD from the Technische Universität München in 2007. In 2011 he was appointed professor of the Chair of Computational Modeling and Simulation at the Technische Universität München. His research interests range from developing methods for simulating and visualizing construction processes over the advancement of Building Information Modeling technologies to developing techniques for simulating pedestrian dynamics. He has co-authored a large number of journal and conference publications in these fields. His email address is: andre.borrmann@tum.de