003-0572

Hammock Activities in Project Scheduling

Sixteenth Annual Conference of POMS,

Chicago, IL, April 29 - May 2, 2005.

György Csébfalvi

University of Pécs, Faculty of Business and Economics, Department of Business Informatics

H-7622 Pécs, Rákóczi út 80, Hungary

E-mail: cseb@ktk.pte.hu, Phone: +36 72 211 433, Fax: +36 72 501 553

Anikó Csébfalvi

University of Pécs, Faculty of Engineering, Department of Structural Engineering

H-7624 Pécs, Boszorkány út 2, Hungary

E-mail: csebfalv@garfield.pmmf.hu, Phone: +36 72 328 684, Fax: +36 72 214 682

Abstract

The concept of hammock activities plays a central role in project management. They are used to fill the time span between other "normal" activities since their duration cannot be calculated or estimated at the initial stage of project planning. However, the recent literature does not offer a general and useful method to compute the unconstrained (resource constrained) duration of such activities. In the proposed approach, a hammock activity is characterized by two dummy activities; therefore the estimation of the unconstrained hammock duration can be formulated as a simple linear programming (LP) problem. The resource-constrained hammock activity duration computation can be described as a mixed integer linear programming (MILP) problem with big-M constraints, which can be solved directly for small-scale projects in reasonable time. The presented implicit enumeration algorithm for the resource constrained hammock activity duration is formulated as a tree-search problem with effective pruning rules. The first pruning rule is based on a special consistency check, which can help to visibilize the "invisible" inconsistencies. The second pruning rule eliminates schedules from explicit enumeration that are known to be unnecessary. The third pruning rule is based on the relaxation of a MILP model, which is a tighter reformulation of the traditional zero-one resource constrained project scheduling model. According to the NP-hard nature of the problem, the proposed implicit enumeration algorithm provides exact solutions for small to medium size problems in reasonable time. Large-scale problems can be managed by introducing an optimality tolerance. In order to illustrate the essence and viability of the proposed new approach, we present detailed computational results for a simple example.

1. Introduction

The concept of hammock activities plays a central role in project management. They are used to fill the time span between other "normal" activities since their duration cannot be calculated or estimated at the initial stage of project planning. Typically, they have been used to denote usage of equipment needed for a particular subset of activities without predetermining the estimated time the equipment must be present on site. Over the past few years the use of hammocks has become popular and most computer software on project scheduling - in the unconstrained case - can now treat them as a part of the whole project analysis process. Nonetheless, some confusion still exists among hammock users, related to the procedure that must be used to calculate their durations after the normal time analysis is performed.

In the unconstrained case, Harhalakis (1990) proposed the first rigorous algorithm to calculate the hammock durations. However, the recent literature does not offer a general and useful method to estimate the hammock durations in the resource-constrained case. The paper presents a new exact approach to cope with this problem.

In order to model hammock activities in projects, we consider the following resource constrained project-scheduling problem: A single project consists of N real activities $i \in \{1, 2, \ldots, N\}$ with a nonpreemptable duration of D_i periods. The activities are interrelated by precedence and resource constraints:

Precedence constraints - as known from traditional CPM-analysis - force an activity not to be started before all its predecessors are finished. These are given by relations $i \rightarrow j$, where $i \rightarrow j$ means that activity j cannot start before activity i is completed. Furthermore, activity $i = 0 \,(i = N + 1)$ is defined to be the unique dummy source (sink). Let $IP_i, i \in \{1, \ldots, N+1\}$ denote the set of immediate predecessors for activity i.

Resource constraints arise as follows: In order to be processed, activity i requires R_{ir} units of resource type $r \in \{1,\dots,R\}$ during every period of its duration. Since resource r, $r \in \{1,\dots,R\}$, is only available with the constant period availability of R_r units for each period, activities might not be scheduled at their earliest (network-feasible) start time but later.

Let T denote the project's makespan and let $T+1$ denote the start time of the unique dummy sink. The traditional approach minimizes the starting time of the unique sink and thus the makespan of the project. In this paper, without loss of generality, we assume that makespan T is the resource-constrained minimal makespan and fix the position of the unique dummy sink in period $T+1$.

Let S_i, $ES_i \le S_i \le LS_i$, denote the start time of activity i, for $i \in \{1,\dots,N\}$, where ES_i (LS_i) denotes the earliest (latest) starting time of activity i in the unconstrained case. Because preemption is not allowed, the ordered set $S = \{S_1,\dots,S_N\}$ defines a schedule of the project.

Let $PS = \{i \to j \mid i \ne j,\ i \in \{1,\dots,N\}, j \in \{1,\dots,N\}\}$ denote the set of predecessor-successor relations. A schedule is network-feasible if satisfies the predecessor-successor relations:

$$S_i + D_i \le S_j, \text{ if } i \to j \in PS. \tag{1}$$

Let $\Re$ denote the set of network-feasible schedules. For a network feasible schedule $S \subset \Re$, let $A_t = \{i \mid S_i \le t < S_i + D_i\}, t \in \{1,\dots,T\}$ denote the set of active (working) activities in period t and let

$$U_{tr} = \sum_{i \in A_t} r_{ir},\ t \in \{1,\dots,T\},\ r \in \{1,\dots,R\} \tag{2}$$

be the amount of resource r used in period t.

A network-feasible schedule $S \subset \Re$ is resource-feasible if satisfies the resource constraints:

$$U_{tr} \le R_r,\ t \in \{1,\dots,T\},\ r \in \{1,\dots,R\}. \tag{3}$$

Let $\overline{\Re} \subseteq \Re$ denote the set of resource-feasible schedules.

Let H denote the number of hammock activities. A hammock activity $\ddot{H}_h, h \in \{1,\dots,H\}$ can be represented by a dummy activity pair with zero duration:

$$\ddot{H}_h = \left\{ i_h \leftrightarrow j_h \mid i_h \in \{1,\ldots,N\}, j_h \in \{1,\ldots,N\}, i_h \neq j_h, D_{i_h} = 0, D_{j_h} = 0 \right\} \quad (4)$$

Let $\ddot{D}_h, h \in \{1,\ldots,H\}$ denote the hammock activity duration. Each $i_h \leftrightarrow j_h, h \in \{1,\ldots,H\}$ dummy activity pair defines a subset of "normal" activities, which has a common start (i_h) and a common end point (j_h). In other words, a dummy activity pair defines the left and right hanging up points of the corresponding "hammock". In general, the duration of a hammock activity is equal to the longest path from the start point to the end point in the corresponding activity subset. Consider a simple resource-constrained example with eighteen "normal" and one hammock activities. The activities are numbered 1 through 20 (plus the dummy activities 0 and 21). The left (right) hanging up point of the hammock is defined by dummy activity 2 (17). There is only one resource type and eleven units are available from the resource type. Without loss of generality, we assume that we know the minimal resource feasible makespan, which is in this example $T = 18$, so we fix the position of the dummy sink in period 19. Table 1 and Figure 1 illustrate the essence of the example.

In Figure 1, the presented network-feasible schedule is not resource feasible, because there are over-utilization in period 11 and 12.

In Figure 1, the activities are represented by bars, the network relations by lines. The unique dummy source (sink) is represented by the $>(<)$ symbol.

The hammock hanging up points are represented by small dark circles, the distance of the hanging up points is illustrated by a gray left-right arrow ($\leftrightarrow$). For the sake of simplicity an absolute time scale is being used in this example. The time periods are labeled by consecutive $t \in \{0,1,\ldots,T,T+1\}$ integers.

Note the convention of starting an activity at the beginning of a time period and finishing it at the end of it. (According to the applied convention, time period one is the first working period.)

Table 1. A simple resource-constrained example

i	D_i	ES_i	LS_i	R_{i1}	IP_i
0	1	0	0		
1	2	1	4	3	{0}
2	0	5	8	0	{3, 4}
3	3	1	1	5	{0}
4	2	3	6	6	{1}
5	3	5	9	5	{2}
6	5	5	8	3	{2}
7	4	4	4	2	{3}
8	3	5	9	1	{4}
9	3	8	12	6	{5}
10	2	10	13	5	{6}
11	2	8	8	2	{7}
12	3	8	12	1	{8}
13	2	11	15	2	{9}
14	2	12	15	3	{10}
15	3	10	10	1	{11}
16	2	11	15	3	{12}
17	0	14	17	0	{13, 14}
18	2	13	13	2	{15}
19	2	14	17	6	{16, 17}
20	4	15	15	5	{18}
21	1	19	19		{19, 20}
	R_1			**11**	

This paper is organized as follows. In Section 2, we show that the calculation of the unconstrained hammock durations can be formulated as a simple linear programming (LP) problem. In Section 3, we present a mixed integer linear programming (MILP) model and an implicit enumeration (IE) algorithm for the computation of resource-constrained hammock durations. Implementation details are presented in Section 4. The last section lists some issues that call for further investigation.

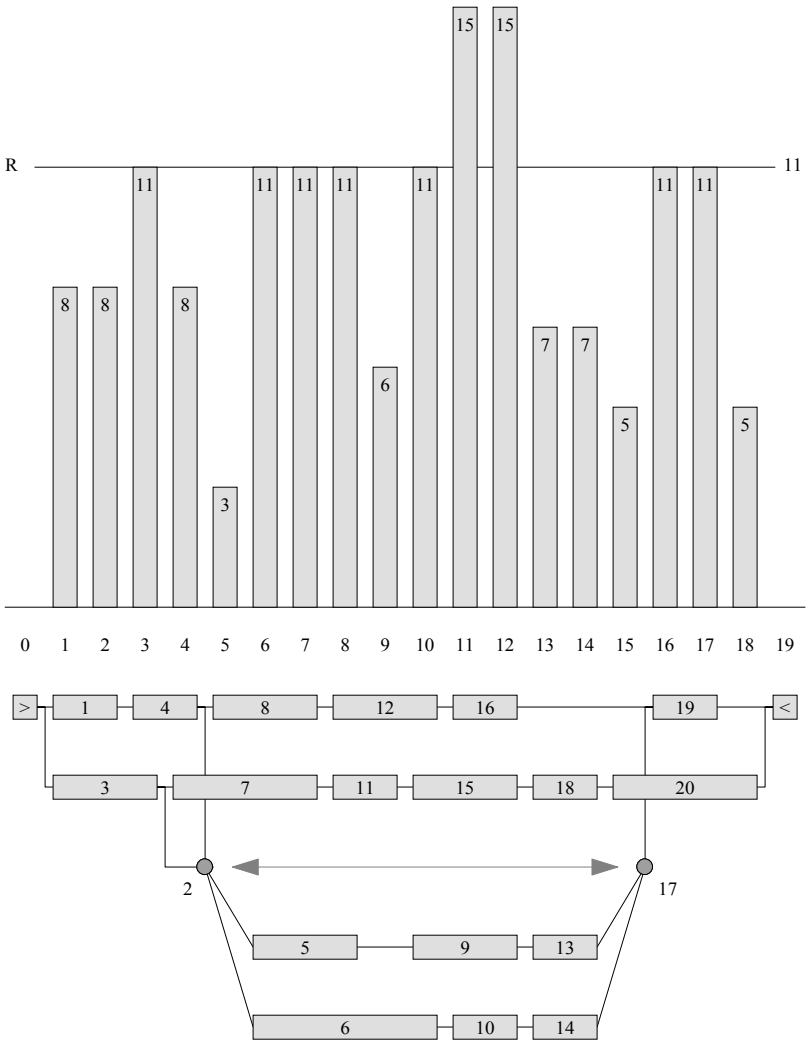


Figure 1. A simple resource-constrained example

2. Unconstrained Hammock Durations

The computation of the unconstrained hammock durations can be formulated as a very simple LP problem. We find a network-feasible $S^* \subset \Re$ schedule, for which the sum of the hammock durations (HD) is minimal:

$$min\left\{HD = \sum_{h=1}^{H} \vec{D}_h = \sum_{h=1}^{H} \left(S_{j_h} - S_{i_h}\right) \mid S \in \Re\right\} = HD^* \tag{5}$$

Figure 2 illustrates the essence of the problem, which, as determined by the well-known software package CPLEX 8.1 (called from MPL modeling environment), has solution $HD^* = 9$. so we can replace the corresponding activity subset $\{5, 9, 13, 6, 10, 14\}$ with a single hammock activity. The result will be an "upper-level" network, which preserve the information on precedence relations and activity durations. This "upper-level" network can be investigated by the traditional *CPM*-analysis. The "upper-level" network is illustrated in Figure 3. Figure 4 show the early and late schedules of the "low-level" network. The "low-level" network can be investigated by an "augmented formulation", in which the additional constraint $HD = HD^*$ describes the hammock duration.

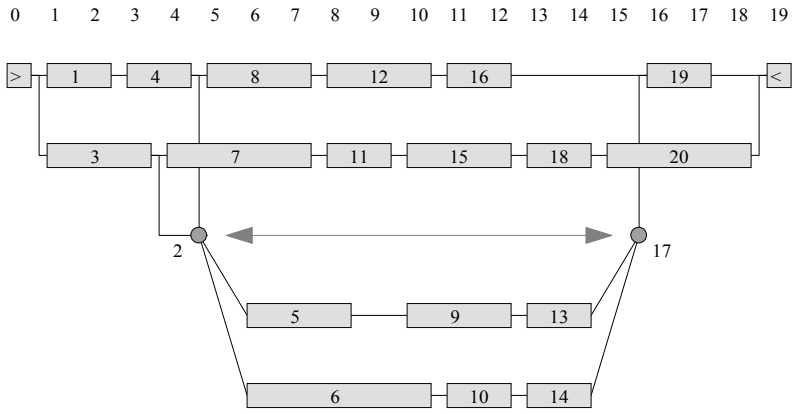


Figure 2. A simple unconstrained example

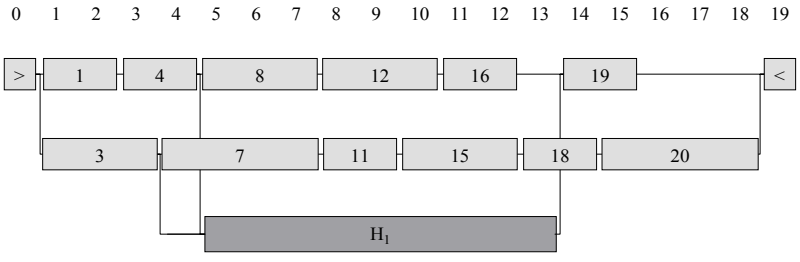


Figure 3. A simple unconstrained example (the upper-level network)

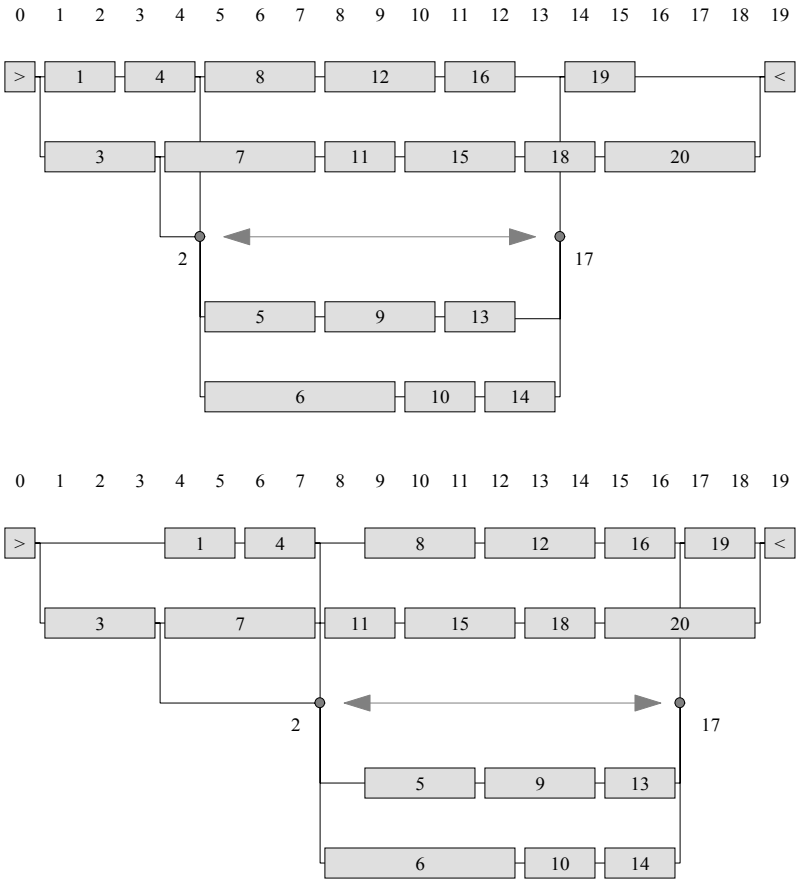


Figure 4. A simple unconstrained example (the low-level early and late schedules)

3. Resource-constrained Hammock Durations

In this section, we present MILP models and an IE algorithm for the computation of resource-constrained hammock durations. The second MILP formulation and the IE algorithm are based on the forbidden (resource constraint violating) set concept. A forbidden activity set F is identified such that: (1) all activities in the set may be executed concurrently, (2) the usage of some resource by these activities exceeds the resource availability, and (3) the set does not contain another forbidden set as a proper subset. See, for example, Bell and Park (1990). A resource conflict can be repaired explicitly by inserting a network feasible precedence relation

between two forbidden set members, which will guarantee that not all members of forbidden set can be executed concurrently. An inserted explicit conflict repairing relation (as its side effect) might be able to repair one or more other conflicts implicitly, at the same time. Let $i \to \cdots \to j$ denote that activity j is a direct (indirect) successor of activity i. An $i \to j$ explicit repairing relation might be replaced by a $p \to q$ implicit relation, where $i \to \cdots \to p$ and $q \to \cdots \to j$, $i \neq p \vee q \neq j$, if there is an other forbidden set for which $p \to q$ is an explicit repairing relation. Let $ER(F)(IR(F))$ denote the set of implicit (explicit) repairing relations for forbidden set F. Figure 9 shows the early *CPM* schedule of our simple example.

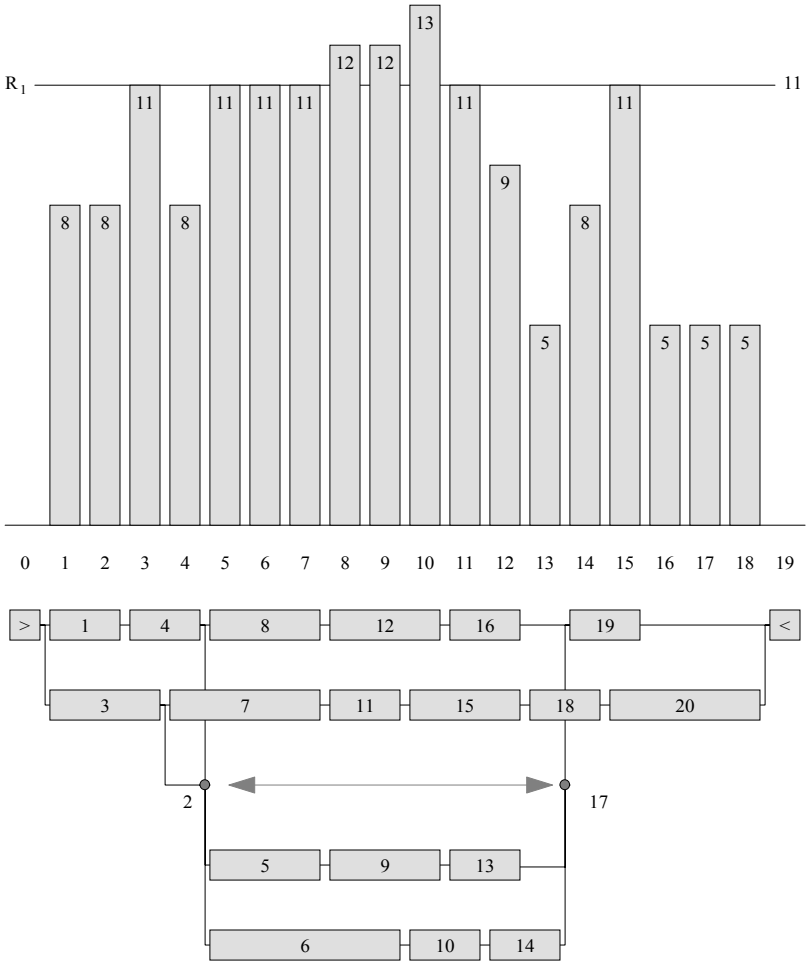


Figure 5. A simple resource-constrained example (the early *CPM* schedule)

Table 2 (3) shows the forbidden sets and their explicit (implicit) repairing sets of the example. In the presented earliest *CPM* schedule every conflict is feasible. Note that a feasible conflict may be "visible" or "hidden". A hidden conflict is "invisible" in the earliest *CPM* schedule, but might be visible in a shifted schedule. In our example, the total number of forbidden sets is sixteen, but in the early schedule only three conflicts - namely $\{F_1, F_{13}, F_{14}\}$ - are visible (active).

Table 2. Forbidden Sets and Explicit Repairs

i	*V*	*Interval*	FS_i	$ER(FS_i)$
1	*V*	[08,09]	{6, 9, 11, 12}	{6→9, 6→12, 11→9, 9→12, 12→9, 11→12}
2		[11,11]	{5, 10, 16}	{5→10, 5→16, 10→16, 16→10}
3		[12,14]	{9, 14, 16}	{9→14, 9→16, 14→16, 16→14}
4		[15,16]	{13, 14, 16, 20}	{13→14, 14→13, 13→16, 16→13, 13→20, 14→16, 16→14, 14→20, 16→20}
5		[08,09]	{6, 8, 9, 11}	{8→6, 6→9, 8→9, 8→11, 11→9}
6		[13,14]	{10, 13, 16, 18}	{10→13, 13→10, 10→16, 16→10, 10→18, 13→16, 16→13, 13→18, 18→13, 16→18, 18→16}
7		[11,11]	{5, 6, 15, 16}	{5→6, 5→15, 5→16, 6→15, 6→16, 15→16}
8		[11,12]	{6, 9, 16}	{6→9, 6→16, 9→16}
9		[13,14]	{9, 10, 18}	{9→10, 10→9, 9→18, 10→18}
10		[11,14]	{9, 10, 16}	{9→10, 10→9, 9→16, 10→16, 16→10}
11		[10,11]	{5, 8, 10, 15}	{5→8, 8→5, 5→10, 5→15, 8→10, 8→15, 15→10}
12		[13,14]	{9, 12, 14, 18}	{9→12, 12→9, 9→14, 9→18, 12→14, 12→18, 18→14}
13	*V*	[10,12]	{9, 10, 15}	{9→10, 10→9, 15→10}
14	*V*	[10,14]	{9, 10, 12}	{9→10, 10→9, 9→12, 12→9, 10→12, 12→10}
15		[10,11]	{5, 10, 12, 15}	{5→10, 5→12, 5→15, 10→12, 12→10, 15→10}
16		[10,11]	{8, 9, 10}	{8→9, 8→10, 9→10, 10→9}

Table 3. Forbidden Sets and Implicit Repairs

i	*V*	*Interval*	FS_i	$IR(F_i)$
1	*V*	[08,09]	{6, 9, 11, 12}	{10→9, 10→12}
2		[11,11]	{5, 10, 16}	{13→10, 5→6, 9→10, 9→12, 9→16, 13→16, 5→8, 5→12, 14→16, 10→12}
3		[12,14]	{9, 14, 16}	{13→14, 13→10, 9→10, 9→12, 13→16, 16→10}
4		[15,16]	{13, 14, 16, 20}	{13→10, 13→18, 16→10, 16→18}
5		[08,09]	{6, 8, 9, 11}	{10→9, 12→9, 8→5}
6		[13,14]	{10, 13, 16, 18}	{14→13, 10→9, 14→16, 10→12}
7		[11,11]	{5, 6, 15, 16}	{9→12, 9→16, 13→16, 5→8, 5→12, 6→12, 10→16, 14→16, 10→12, 18→16}
8		[11,12]	{6, 9, 16}	{10→9, 6→12, 10→16, 14→16, 10→12, 9→12, 13→16}
9		[13,14]	{9, 10, 18}	{13→10, 13→18}
10		[11,14]	{9, 10, 16}	{13→10, 9→12, 13→16, 14→16, 10→12}
11		[10,11]	{5, 8, 10, 15}	{13→10, 5→6, 9→10, 16→10, 8→6, 12→10, 8→11}
12		[13,14]	{9, 12, 14, 18}	{13→14, 13→10, 9→10, 13→18, 16→10, 16→14, 12→10, 16→18}
13	*V*	[10,12]	{9, 10, 15}	{13→10}
14	*V*	[10,14]	{9, 10, 12}	{13→10, 16→10}
15		[10,11]	{5, 10, 12, 15}	{13→10, 5→6, 9→10, 9→12, 5→8, 16→10}
16		[10,11]	{8, 9, 10}	{12→9, 8→5, 16→10, 8→6, 12→10, 13→10}

Note that in the unconstrained case the hammock durations can be calculated uniquely and unambiguously, while in the resource-constrained case the calculation of the hammock durations can be performed following one of two ways.

Let $\vec{C}_h, h \in \{1, \ldots, H\}$ define the per period cost of hammock activities. In the first approach, we apply the traditional "visible conflict" oriented repairing strategy, so we find a resource-feasible $S^* \subset \overline{\Re}$ schedule, for which the resource-constrained total hammock cost (HC) is minimal:

$$min\left\{HC = \sum_{h=1}^{H} \vec{C}_h \vec{D}_h = \sum_{h=1}^{H} \vec{C}_h \left(S_{j_h} - S_{i_h}\right) \mid S \in \overline{\Re}\right\} = HC^* \tag{6}$$

Note that in this approach the optimal solution $S^* \subset \overline{\Re}$ is a schedule (the variables are activity starting times and we describe the resource constraints explicitly), so we know nothing about the resource-feasible activity shifts. In other words, a shifted version of the optimal schedule not necessarily will be resource-feasible.

In the second approach, we replace the traditional "visible conflict" oriented strategy by a "feasible conflict" oriented one. In other words, we repair every feasible resource conflict regardless of whether it is "visible" or "hidden". In this case, the primary variables are conflict repairing relations, so the optimal solution will be a resource-feasible solution set, in which every movable activity can be shifted without effecting the resource feasibility. Note that the resource-feasible scheduling flexibility of the optimal solution can be investigated by the traditional CPM-analysis.

Let $\overline{\overline{\Re}}$ define the set of the feasible conflict repairing sets. Let $FR \subset \overline{\overline{\Re}}$ denote a feasible conflict repairing set. A feasible repairing set $FR \subset \overline{\overline{\Re}}$ is a consistent relation set, which is able to resolve every visible or hidden resource conflict in the given project, so after inserting all the relations of such a set we get a resource-feasible schedule set. In this case, the starting times are

secondary variables: $S = S(FR)$. In the second approach, we find a $CR^* \subset \overline{\overline{\Re}}$ conflict repairing set, for which the resource-constrained total hammock cost (HC) is minimal:

$$min\left\{HC = \sum_{h=1}^{H} \ddot{C}_h \, \vec{D}_h = \sum_{h=1}^{H} \ddot{C}_h \left(S_{j_h} - S_{i_h} \right) \middle| FR \subset \overline{\overline{\Re}} \right\} = HC^* \tag{7}$$

In the second case we describe the resource constraints implicitly. Note that the application of either approach may give totally different hammock durations in the function of the current $\ddot{C}_h, h \in \{1, \ldots, H\}$ costs, when $H > 1$. Applying the second model, we can answer several "what if" like questions. For example, from managerial point of view, may be interesting to know which schedule set gives the maximal hammock cost, because a higher hammock cost may be compensated by a larger scheduling flexibility. Let us denote with $PR = \{PR_1, PR_2, \ldots\}$ the set of possible conflict repairing relations. In our simple problem $|PR| = 50$, the possible conflict repairing relations are shown in Table 4. A feasible conflict repairing set FR is a subset of PR: $FR \subseteq PR$.

Table 4.

i	FR_i	i	FR_i
1	6→9	26	13→10
2	6→12	27	10→18
3	11→9	28	13→18
4	9→12	29	18→13
5	12→9	30	16→18
6	11→12	31	18→16
7	5→10	32	5→6
8	5→16	33	5→15
9	10→16	34	6→15
10	16→10	35	6→16
11	9→14	36	15→16
12	9→16	37	9→10
13	14→16	38	10→9
14	16→14	39	9→18
15	13→14	40	5→8
16	14→13	41	8→5
17	13→16	42	8→10
18	16→13	43	8→15
19	13→20	44	15→10
20	14→20	45	12→14
21	16→20	46	12→18
22	8→6	47	18→14
23	8→9	48	10→12
24	8→11	49	12→10
25	10→13	50	5→12

3.1 MILP formulations

The first model for the resource-constrained hammock cost is a simple modification of the traditional resource-constrained MILP model:

$$min\left[HC = \sum_{h=1}^{H} \vec{C}_h \ \vec{D}_h = \sum_{h=1}^{H} \vec{C}_h \left(S_{j_h} - S_{i_h} \right) \right] = HC^* \tag{8}$$

subject to

$$\sum_{t=ES_i}^{LS_i} S_{it} = 1, i \in \{1,\ 2, \ldots, N\} \tag{9}$$

$$S_i = \sum_{t=ES_i}^{LS_i} t * S_{it}, i \in \{1,\ 2, \ldots, N\} \tag{10}$$

$$\sum_{t=ES_j}^{LS_j} t * S_{jt} \geq \sum_{t=ES_i}^{LS_i} t * S_{it} + D_i, i \rightarrow j \in PS \tag{11}$$

$$\sum_{i=1}^{N} R_{ir} \ * \ \sum_{s=t-D_i+1}^{t} S_{is} \leq R_r, r \in \{1,\ 2, \ldots, R\}, t \in \{1,\ 2, \ldots, T\} \tag{12}$$

$$S_{it} \in \{0,\ 1\},\ i \in \{1,\ 2, \ldots, N\}, t \in \{ES_i, \ldots, LS_i\} \tag{13}$$

The objective function (8) minimizes the total hammock cost. Constraint set (9) assures that to each activity a unique start time within its time window is assigned. Constraint set (10) describes the relation between the integer and binary start time variables. Constraints (11) take into consideration the precedence relations between each pair of activities $i \rightarrow j$, where i immediately precedes j. Finally, constraint set (12) limits the total resource usage within each period to the available amount.

Using the first model, the optimal solution of our example is $HC^* = 9$. The optimal schedule is presented in Figure 5. Unfortunately, this model is unable to give information about the resource-

constrained activity shifts (some delay may be able to destroy the resource-feasibility), so the practical importance of the first model is limited. Naturally, the application of either model to calculate the hammock cost yield the same total hammock cost, but the second model gives additional information about the scheduling flexibility.

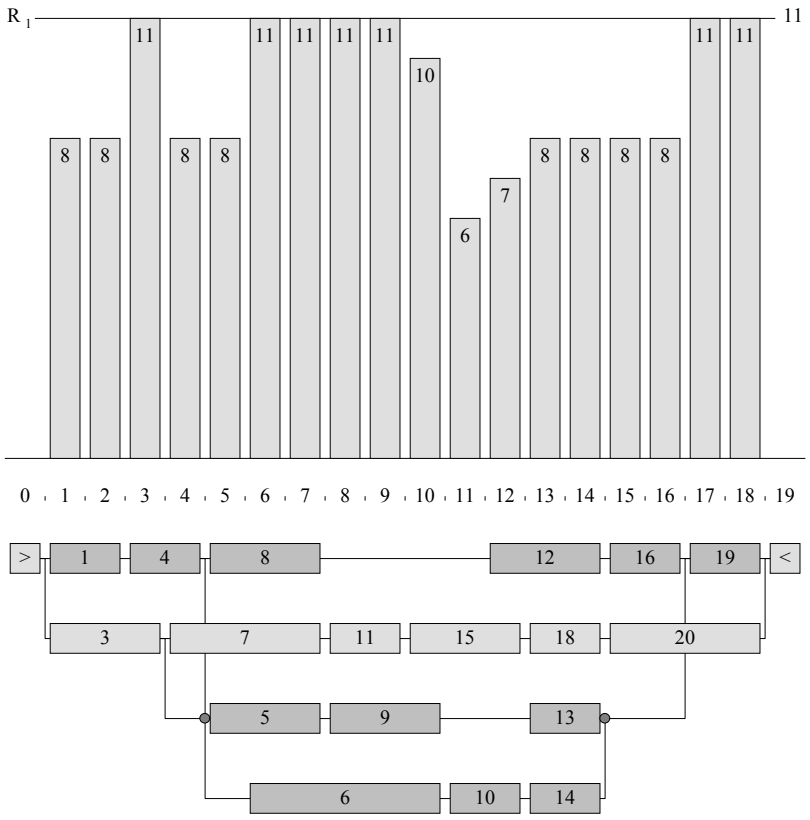


Figure 6. A simple resource-constrained example (the optimal hammock duration using the first model)

The second model is based on the forbidden set concept. In this model the total number of zero-one variables is $|PR|$, and the formulation is based on well-known "big-M" constraints. Defining the decision variables

$$Y_{ij} = \begin{cases} 1 & if\ \ i \to j\ \ inserted \\ 0 & otherwise \end{cases} \quad ,where\ \ i \to j \in PR\,, \tag{14}$$

the following MILP model arises:

$$min\left[HC=\sum_{h=1}^{H}\vec{C}_h\ \vec{D}_h=\sum_{h=1}^{H}\vec{C}_h\left(S_{j_h}-S_{i_h}\right)\right]=HC^* \tag{15}$$

subject to

$$\sum_{i\to j\in S_f} Y_{ij}\geq 1,\ \text{where}\ S_f=ER\left(F_f\right)\cup IR\left(F_f\right),\ f\in\left\{1,\ldots,\left|FS\right|\right\} \tag{16}$$

$$\underline{S}_i+D_i\leq S_j+\left(LS_i-ES_j+D_i\right)*\left(1-Y_{ij}\right),\ i\to j\in PR \tag{17}$$

$$S_i+D_i\leq S_j,\ i\to j\in PS \tag{18}$$

$$Y_{ij}\in\{0,1\},\ \text{for every}\ i\to j\in PR. \tag{19}$$

The objective function (15) minimizes the proposed resource constrained total hammock cost. Constraint set (16) assures the resource feasibility (we have to repair each resource conflict explicitly or implicitly, therefore from each conflict repairing set we must choose at least one element).

Constraint sets (17) take into consideration the precedence relations between activities in the function of repairing relations. In constraint sets (17) $LS_i\ (ES_j)$ denotes the latest (earliest) start time of activity $i\,(j)$ in the network-feasible earliest (latest) CPM schedule. Note that the $(LS_i-ES_j+D_i)$ values are optimal (minimal) "big-M" constants in the constraint set (17).

Constraint sets (18) take into consideration the original precedence relations between activities.

The optimal schedule of the second model is presented in Figure 5. Easy to see, that the additional conflict repairing relations destroy the original structure of the project.

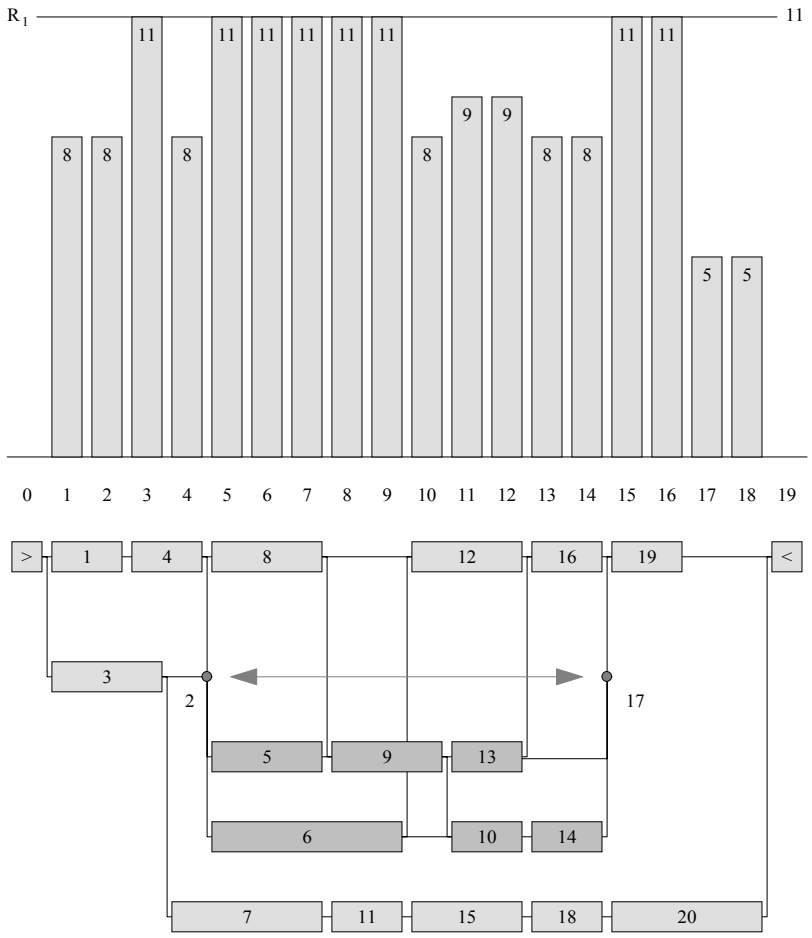


Figure 7. A simple resource-constrained example (the optimal hammock duration using the second model)

3.2 An Implicit Enumeration Algorithm

In this section we present a new exact implicit enumeration algorithm for the computation of the resource-constrained total hammock cost. The crucial point of such a development is the constraining power of the applied lower bounding technique. Without tight lower bounds the size of the search tree would be extremely large. The "big-M" formulation applied in the second MILP approach is the simplest way to model the resource feasibility, but it does suffer, unfortunately, from a weak LP relaxation as any other "big-M" like formulation does. The IE

algorithm is formulated as a tree-search problem with three effective pruning rules. The first pruning rule is based on a special consistency check, which can help to visibilize the "invisible" inconsistencies. The second pruning rule eliminates schedules from explicit enumeration that are known to be unnecessary. The third pruning rule is based on the relaxation of a MILP model, which is a tighter reformulation of the traditional zero-one resource constrained project scheduling model. To solve the relaxed problems a fast "state-of-the-art" interior point solver (BPMPD) was used. According to the NP-hard nature of the problem, the proposed IE algorithm provides exact solutions for small to medium size problems in reasonable time. Large-scale problems can be managed by introducing an optimality tolerance. Use of optimality tolerance drastically decreases the size of the searching tree. In order to illustrate the essence and viability of the proposed new algorithm, we present detailed computational results for our simple example.

The proposed lower bounding technique is based on the forbidden set concept. Let $FS = \left\{ FS_f = \left\{ F_{fi} \,\middle|\, i \in \left\{ 1, 2, \ldots, \left| FS_f \right| \right\}, F_{fi} \in \left\{ 1,\ 2, \ldots, N \right\} \right\} \,\middle|\, f \in \left\{ 1, 2, \ldots, \left| FS \right| \right\} \right\}$ denote the set of the feasible forbidden sets of the project. Our objective is to repair every feasible resource conflict such that the proposed HC measure is minimized.

In order to develop an appropriate lower bounding technique, let S_{it}, where $ES_i \le t \le LS_i$, denote a zero-one decision variable:

$$S_{it} = \begin{cases} 1 & \textit{if activity } i \textit{ is started in period } t \\ 0 & \textit{otherwise} \end{cases}, \; i \in \{1, \ldots, N\}. \tag{20}$$

According to the applied notation:

$$S_i = \sum_{t=ES_i}^{LS_i} t * S_{it}, \;\; \sum_{t=ES_i}^{LS_i} S_{it} = 1, \; i \in \{1, \ldots, N\}. \tag{21}$$

The traditional zero-one resource-constrained project scheduling model with the new objective is the following:

$$min\left[HC = \sum_{h=1}^{H} \vec{C}_h \ \vec{D}_h = \sum_{h=1}^{H} \vec{C}_h \left(S_{j_h} - S_{i_h} \right) \right] = HC^* \tag{22}$$

subject to

$$\sum_{t=ES_i}^{LS_i} S_{it} = 1, i \in \{1, 2, \ldots, N\} \tag{23}$$

$$S_i = \sum_{t=ES_i}^{LS_i} t * S_{it}, i \in \{1, 2, \ldots, N\} \tag{24}$$

$$\sum_{t=ES_j}^{LS_j} t * S_{jt} \geq \sum_{t=ES_i}^{LS_i} t * S_{it} + D_i, i \rightarrow j \in PS \tag{25}$$

$$\sum_{i=1}^{N} R_{ir} * \sum_{s=t-D_i+1}^{t} S_{is} \leq R_r, r \in \{1, 2, \ldots, R\}, t \in \{1, 2, \ldots, T\} \tag{26}$$

$$S_{it} \in \{0, 1\}, \ i \in \{1, 2, \ldots, N\}, t \in \{ES_i, \ldots, LS_i\} \tag{27}$$

The objective function (22) minimizes the total hammock cost. Constraint set (23) assures that to each activity a unique start time within its time window is assigned. Constraints (24) describe the relation between the integer and binary start time variables. Constraints (25) take into consideration the precedence relations between each pair of activities $i \rightarrow j$, where i immediately precedes j. Finally, constraint set (26) limits the total resource usage within each period to the available amount. We can replace the traditional precedence constraint set (25) with a totally unimodular formulation:

$$\sum_{s=t}^{LS_i} S_{is} + \sum_{s=ES_j}^{t+D_i-1} S_{js} \leq 1, \ i \rightarrow j \in PS, \ t \in \{ES_j - D_i + 1, \ldots, LS_i\} \tag{28}$$

Constraint set (28) assures that activity j must not be begun before time $t + D_i$ if activity i is started at time t or later. Note that the LP relaxation (22)-(24), (26), and (28) is stronger then (22)-(26). See, for example, Demeulemeester and Herroelen (2002).

We will now show that the traditional resource constraint set (26) can be replaced by a new forbidden set oriented formulation. Let $\overleftarrow{P}_f \left(\vec{P}_f\right)$ denote the first (last) time period in which forbidden set FS_f, $f \in \{1, 2, \ldots, |FS|\}$ may be active (visible):

$$\overleftarrow{P}_f = \max\left\{ES_i \,\middle|\, i \in FS_f\right\} \tag{29}$$

$$\vec{P}_f = \min\left\{LS_i + D_i - 1 \,\middle|\, i \in FS_f\right\}$$

The forbidden set oriented formulation can be described as follows:

$$\sum_{i \in FS_f} \sum_{s=t-D_i+1}^{t} S_{is} \le |FS_f| - 1, r \in \{1, 2, \ldots, R\}, \; t \in \left\{\overleftarrow{P}_f, \ldots, \vec{P}_f\right\}, \; f \in \{1, 2, \ldots, |FS|\} \tag{30}$$

Constraints (30), according to the definition, simply describe the fact that the concurrent execution of the forbidden set members is prohibited in every affected time period. Note that the LP relaxation (22)-(24), (26), and (28) may be weaker or stronger than (22)-(24), (28), and (30). Therefore, the LP relaxation (22)-(24), (26), (28), and (30) will be at least as strong as (22)-(24), (26), and (28) or (22)-(24), (28), and (30). Theoretically, (22)-(24), (26), (28), and (30) is a redundant MILP model, in which either (26) or (30) is not necessary, but the redundant constraints, as valid cuts, greatly strengthen the LP relaxation of the model.

Relaxing the integrality assumption $\left(S_{it} \in \{0, 1\} \Rightarrow S_{it} \in [0, 1]\right)$, we get an LP problem, which - using a fast interior point solver - can be solved in reasonable time. When we solve the relaxed minimization problem, we get a lower bound for HC :

$$\underline{HC} = \sum_{h=1}^{H} \vec{C}_h \left(S^*_{j_h} - S^*_{i_h}\right) \tag{31}$$

The LP relaxation provides good quality lower bounds for the total hammock cost, which is essential in a tree search process. According to the progress of the tree search process, the

schedules become more and more resource constrained. The more constrained a schedule, the smaller the gap between the estimated and the true lower bounds.

The tree-building process is based on the forbidden set concept. The nodes of the tree correspond to "partial" schedules. In our IE algorithm, any partial schedule satisfies all original precedence constraints and assigns a start time to all activities. But it is "partial" because it may violate one or more "visible" or "hidden" resource constraints. The nodes are characterized by the non-redundant subset of the original network relations and the additional conflict repairing relations, the feasible subset of the original forbidden sets, the visible subset of the feasible subset, and the precedence but not necessarily resource feasible earliest (latest) starting times:

$$\left\{ FR^{(n)}, FS^{(n)}, VS^{(n)}, \left\{ \left\{ ES_i^{(n)}, LS^{(n)} \right\} \middle| \; i \in \{1, 2, \ldots, N\} \right\} \right\} \tag{32}$$

where $FR^{(n)}$ denotes the set of the inserted conflict repairing relations, $FS^{(n)} \subseteq FS^{(0)} = FS$, and $VS^{(n)}$ denotes the visible subset, $VS^{(n)} \subseteq FS^{(n)}$. Leaf nodes of search tree are resource feasible or pruned schedules. Our node evaluation (fitness) function is very simple: It assigns to each $n \in \{0, 1, 2, \ldots\}$ node the estimated $\underline{HC}^{(n)}$ lower bound value. Thus, at each step of the tree-building process, we select the most promising node, which has been generated but not expanded. A parent node is transformed into a set of child nodes by repairing its "best" resource conflict all the possible ways. Note that, in this context, "best" means a conflict with minimal number of possible repairing relations. According to our "best-first" searching strategy, a node without feasible resource conflicts will be a solution of the total hammock cost minimization problem. Note that an inserted explicit conflict repairing relation (as its side effect) may be able to repair one or more other conflicts implicitly, at the same time.

In the traditional forbidden set oriented problem solving strategy a parent node is transformed into a set of child nodes by repairing its first resource conflict all the possible ways, where "first"

always means the earliest conflict in time interval $[l,T]$. The reason is very simple: in the traditional case we would like to get a resource feasible solution as early as possible and after that the searching process terminates.

In our case: (1) we have to generate all the solutions of the problem, (2) the first resource-constrained solution will not necessarily be optimal for HC, so we have to find other solutions (when we use the modified "best" conflict repairing strategy, the tree usually will be smaller than the traditional tree), and (3) the total hammock cost is not a regular measure of performance, therefore the application of a traditional regular pruning rule may "over-prune" the searching tree. So we have to develop special pruning rules to cut down the effective branching factor of the search tree. In this study we applied three pruning rules which are able to substantially reduce the number of generated nodes.

(1) The first "cyclic repairs rule" is based on a special consistency check, which can help to visibilize the "invisible" inconsistencies. The basic idea of this rule is very simple: After inserting a conflict repairing relation and updating the schedule, our tree-building process "looks ahead" and in a cyclically repeatable repairing process repairs each resource conflict in the child node which has exactly one repairing possibility. This cyclical repairing process immediately terminates and the child node is discarded if one ore more conflicts become non-repairable or the updated project duration exceeds the prescribed maximal project duration.

(2) The second "at least as shiftable rule" is a straightforward modification of the well-known "left shiftable rule" which is an efficient regular pruning rule. Let $MS^{(n)}, n \in \{0, 1, 2, ...\}$ denote

the non-redundant subset of the predecessor-successor relations for the movable (non-critical) activities:

$$MS^{(n)} = \left\{ i \to j \,\middle|\, i \to j \in NonRed\left\{ PS \cup FR^{(n)} \right\}, ES^{(i)} < LS^{(i)}, ES^{(j)} < LS^{(j)} \right\} \qquad (33)$$

The applied $NonRed\{\cdot\}$ operator eliminates the redundant predecessor-successor relations, for example, $NonRed\{ i \to j,\ j \to k,\ i \to k \} \Rightarrow \{ i \to j,\ j \to k \}$. The modified rule compares two nodes: If $MS^{(a)} \supseteq MS^{(b)}$, $ES^{(a)} \geq ES^{(b)}$, $LS^{(a)} \leq LS^{(b)}$, and $FS^{(a)} = FS^{(b)}$, then node a can be immediately pruned (in other words, node a is dominated by node b).

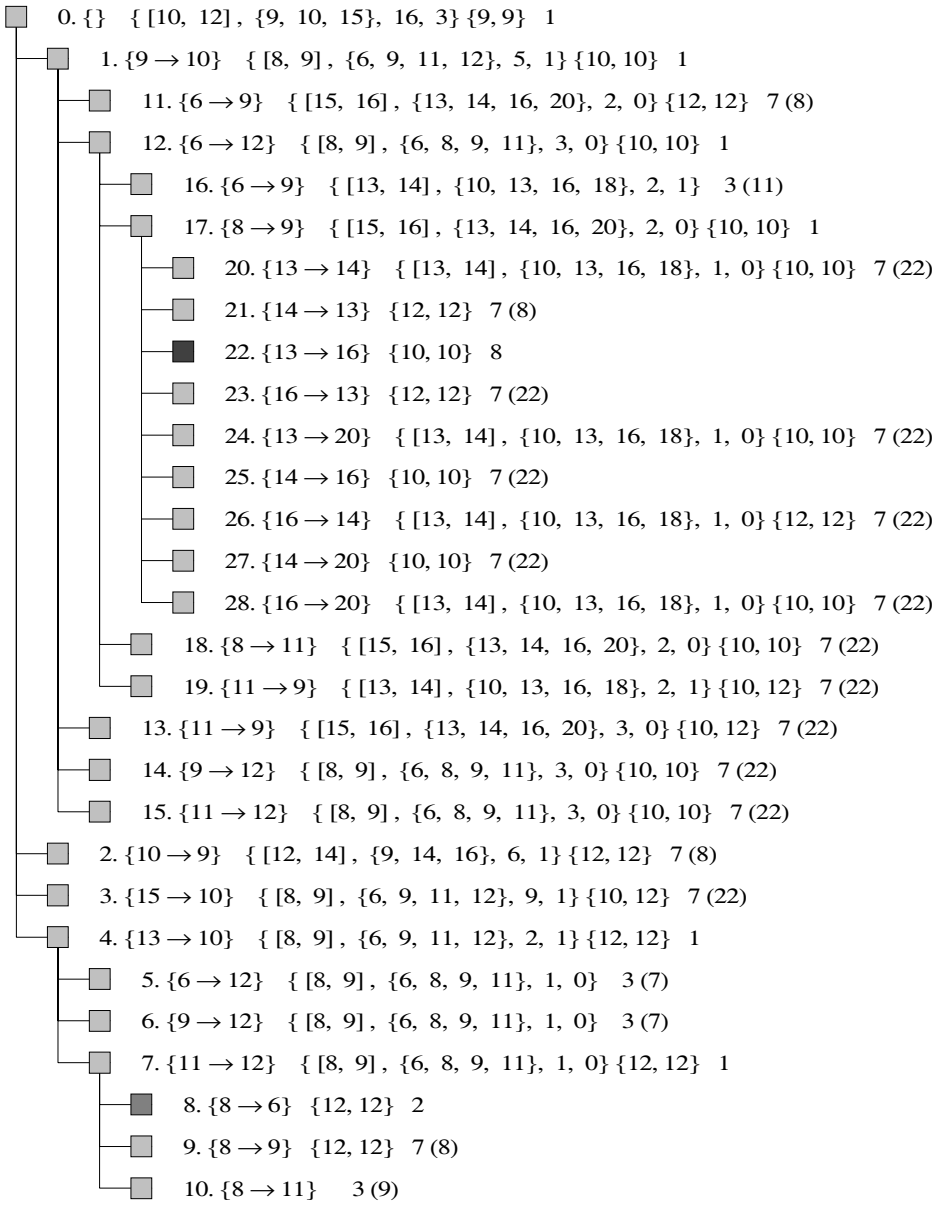
(3) The third rule is based on the relaxation of the proposed new MILP formulation. A child node is discarded if the relaxed LP solution is primal infeasible. In this study, to solve the relaxed LP, a very fast primal-dual interior method (BPMPD) developed by Mészáros (1996) was used.

In the tree building process, the applied new redundant MILP formulation is able to detect the resource unfeasibility earlier than the traditional formulation does, therefore results in a smaller tree.

The algorithm maintains the dynamically changing $\{ Best\ Node, Best\ HC \} = \{ best, HC^{(best)} \}$ set.

A generated child node c can be immediately discarded if $\underline{HC}^{(c)} \geq HC^{(best)}$.

The search tree of our example is shown in Figure 7. Figure 8 illustrate the optimal schedule set for the proposed objective. In the optimal schedule set ($n = 22$) the total hammock cost is ten $\left(\vec{C}_1 = 1\right)$. The size of the search tree is 28. The proposed IE algorithm solved the problem very quickly, the computation time was 0.134 sec.



Legend :

$$Node.\ Repair\ [Cyclic\ Repairs]\left\{\left[\vec{P},\vec{P}\right],Conflict,|FS|,|VS|\right\}\{\ \underline{HC},HC\ \}\ Flag=\begin{cases}0 & Generated\\ 1 & Expanded\\ 2 & Feasible\\ 3 & Pruned\ By\ (\ Node\ )\\ 4 & if\ \ Primal\ Infeasible\ \ [Cyclic\ Conflicts]\\ 5 & Unrepairable\ Set\\ 6 & Unrepairable\ Sets\\ 7 & Pruned\ By\ (\ Best\ )\\ 8 & Optimal\end{cases}$$

Figure 8. A simple resource-constrained example (the search tree)

4. Implementation Details

In this study, as a MILP solver, the popular and very fast "state-of-the-art" CPLEX 8.1 (called from MPL modeling environment) was used. Naturally, this solver can be replaced by any other commercial MILP solver. The automatic MPL input file generator of the proposed model has been programmed in Visual C++® Version 6.0. The generator, as a DLL, was built into the *ProMan* system developed by Ghobadian and Csébfalvi (1995), Csébfalvi (2002). The implicit enumeration algorithm has been programmed in Visual C++® Version 6.0. The algorithm, as a DLL, was built into the *ProMan* system.

The figures (projects and trees) presented in this paper, are Windows® meta-files, which have been generated automatically by the *ProMan* system. In *ProMan* a "Windows-like" tree representation form has been applied. Note that *ProMan* is a mouse-oriented system, so each dummy activity has positive duration to allow drag and drop actions.

To solve the relaxed MILP problems a fast "state-of-the-art" primal-dual interior point solver, namely the BPMPD developed by Mészáros (1996), was used. Naturally, this solver can be replaced by any other commercial (academic) LP solver. The computational results were obtained by running ProMan and MPL (CPLEX 8.1) on a 1.8 GHz Pentium IV IBM PC with 256 MB of memory under Microsoft Windows XP® operation system.

5. Conclusions

In this paper, we presented a new mixed integer linear programming models and an implicit enumeration algorithm for the computation of the hammock durations (cost). In the proposed approach, a hammock activity was characterized by two dummy activities. The calculation of the unconstrained hammock durations was formulated as a simple linear programming (LP) problem. The resource-constrained hammock activity duration computation was described as a mixed integer linear programming (MILP) problem with big-M constraints. The presented implicit enumeration algorithm for the resource constrained hammock activity duration was formulated as a tree-search problem with effective pruning rules, According to the NP-hard nature of the problem, the proposed implicit enumeration algorithm provides exact solutions for small to medium size problems in reasonable time. Large-scale problems can be managed by introducing an optimality tolerance. The obtained results left a margin for at least three interesting improvements:

(1) In the presented new MILP model the objective function can be replaced by any other objective, which can be described as a function of the earliest (latest) starting time variables.

(2). In the proposed algorithm we have to solve LP problems to get a lower bound values. It is an open and very hard question, what would be the "best" big-M free formulation, which would be able to produce tighter lower bounds.

(3) In the node-expanding phase of the algorithm, we applied very simple rules to select the most promising node and conflict. It is a very interesting and challenging question, what would be the "best" selection-expansion strategy, which would be able to produce smaller trees.

Acknowledgement

The author would like to express his thanks to Cs. Mészáros for making the unlimited WIN95/NT DLL version of the BPMPD (`www.sztaki.hu/~meszaros/bpmpd`) interior point solver available.

References

Alvares-Valdés, R. and Tamarit, J. M. (1993) The Project scheduling Polyhedron: Dimension, Facests and Lifting Theorem, European Journal of Operational Research, **67**, 204-220.

Bell, C. E., and Park, K. (1990) Solving Resource-constrained Project Scheduling Problems by A* Search, *Naval Research Logistics*, **37**, 61-84.

Csébfalvi, G. (1998) A fast exact solution procedure for the multiple resource-constrained project scheduling problem, *Proc. APMOD '98 Extended Abstracts*, Limasol, Cyprus.

Csébfalvi, G. (2002) *ProMan Manual (Version 2.1)*, University of Pécs, Faculty of Business and Economics, Hungary.

Demeulemeester, E. L., and Herroelen, W. S. (2002) *Project Scheduling: A Research Handbook*, Kluwer, Boston / Dordrecht / London.

Ghobadian, A., and Csébfalvi, G. (1995) "Workshop on Developing Interactive Learning Material for Project Management," *Proc. of 1995 Annual Meeting of Decision Sciences Institute*, Boston, US.

Harhalakis, G. (1990) Special Features of precedence network charts, European Journal of Operational Research, **49**, 50-59.

Mészáros, Cs. (1996) The Efficient Implementation of Interior Point Methods for Linear Programming and their Applications, *PhD Thesis*, Eötvös Loránd University of Sciences, Hungary.