BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 63, No. 3, 2015
*DOI: 10.1515/bpasts-2015-0072*

# Heuristics for project scheduling with discounted cash flows optimisation

M. KLIMEK[1]* and P. ŁEBKOWSKI[2]

[1] Department of Computer Science, Pope John Paul II State School of Higher Education, 95-97 Sidorska St., 21-500 Biała Podlaska, Poland
[2] AGH University of Science and Technology, Department of Operations Research and Information Technology, 10 Gramatyka St., 30-067 Kraków, Poland

**Abstract.** The article presents the resource-constrained project scheduling problem with the maximisation of discounted cash flows from the contractor's perspective: with cash outflows related to starting individual activities and with cash inflows for completing project stages (milestones). The authors propose algorithms for improving a forward active schedule by iterative one-unit right shifts of activities, taking into account different resource flow networks. To illustrate the algorithms and problem, a numerical example is presented. Finally, the algorithms are tested using standard test problems with additionally defined cash flows and contractual milestones.

**Key words:** resource-constrained project scheduling, discounted cash flows, milestones.

## 1. Introduction

The Resource Constrained Project Scheduling Problem (RCPSP) has been the subject matter of numerous research studies. From the practical perspective, project scheduling supplemented with an analysis of economic effects of the planning decisions made is of utmost importance. An analysis of financial effects is most often carried out taking into consideration changes in the value of money in time, by computing the Net Present Value (NPV) of cash flows at the assumed discount rate. Research into NPV maximising for the project scheduling problem was initiated by Russell in 1970 (the Max-NPV model [1]). Since then, numerous optimisation models for the RCPSP with Discounted Cash Flows (RCPSPDCF) have been considered. Recent results in this field include [2–6]. For a detailed description of to-date research, models and algorithms used for optimising project NPV the reader is referred to review papers, including [7, 8]. This paper discusses selected problems pertaining to the optimisation model analysed.

Net Present Value of project is optimised from the contractor's or customer's (principal's) perspective. In this paper, NPV maximising from the contractor's perspective is studied; in this case, cash inflows (positive cash flows) are the customer's payments to the contractor for tasks performed, while cash outflows (negative cash flows) are the customer's expenses incurred in connection with the execution of the project's activities. The contractor's expenses are, as a rule, more frequent than payments the contractor receives and expense amounts depend on numerous factors, including materials acquisition cost and resources commitment.

The drivers of the project's NPV include the schedule of the customer's payments to the contractor. The research papers [2, 9, 10] analyse the Payment Project Scheduling (PPS) problem, in which there are established rules for financial settlements between the customer and the contractor, that is the aggregate of the customer's payments under the project, number of payment tranches, amounts and deadlines of individual payments. The PPS problem is examined from the contractor's perspective [2, 10], as well as from the principal's perspective [11]. Solutions are also sought for satisfactory for the both parties: the customer and the contractor [12].

Amounts and dates of individual cash flows should take into consideration numerous factors, such as activity execution cost, work progress, project execution progress etc. The customer's and the contractor's expectations in this scope diverge: the contractor is interested in receiving the customer's payments as soon as possible, while the customer would rather delay payments, preferably, the customer would pay only after the complete project execution. The research studies [2, 10] examine various payment models, including:

- Lump-Sum Payment (LSP);
- Payments at Event Occurrences (PEO), e.g., upon the completion of selected milestones or upon the completion of certain activities – Payments at Activities' Completion (PAC) times;
- Equal Time Intervals (ETI) (with a predefined number of payments) or Progress Payments (PP) in time intervals with undefined number of payments made until the project completion time.

Customer-contractor settlement models also include a bonus-penalty system [4–6, 13, 14]. Penalties are imposed for a failure to complete the execution of the project or its milestones by the predefined dates, and bonuses are awarded for work completion ahead of the schedule. Time windows are defined,

*e-mail: m.klimek@dydaktyka.pswbp.pl

there is such time intervals that work completed within them is neither awarded, nor penalised. Models are also analysed with project settlement in milestones for a Multi-Mode RCPSP (MMRCPSP) [15, 16].

In this paper, the authors analyse the project scheduling problem with limited availability of resources, with predefined milestones and a single mode of activity execution: Single-Mode RCPSP. The authors discuss their own optimisation model [4–6, 14] with financial settlements in milestones, where the objective function is Net Present Value maximisation – aggregate discounted cash flows with the customer's payments for milestone completion, a system of penalties for late execution of project milestones and the contractor's outflows related to activity execution.

For the RCPSP, forward scheduling or backward scheduling is used most often, with activity list decoding with use of a serial or parallel SGS (Schedule Generation Scheme) [17]. For the problem considered, forward scheduling is analysed, improvable by right shifts of activities. It is, for instance, a good idea to delay activities whose right shift in time does not delay project milestone completion.

The objective of this paper is to present new iterative right-shift algorithms and to prove their effectiveness for the project settlement in milestones with the maximisation of aggregate DCF. Right shifts are performed with a predefined resource allocation to activities taking into consideration ordering and resource constraints. To illustrate the problem and algorithms, a computation example is presented. Finally, results of computation experiments are analysed for test instances from the Project Scheduling Problem LIBrary (PSPLIB) [18], with additionally defined cash flows and milestones.

## 2. Problem formulation

This paper analyses the Resource Constrained Project Scheduling Problem, where activity (task) pre-emption is not allowed and each activity is executed in a single predefined mode, known as the non-pre-emptive single-mode RCPSP.

For the purposes of formulating the cash flows optimisation problem from the contractor's perspective, the following assumptions have been adopted:

- the contractor's inflows are the customer's payments made exactly at the project milestone completion times;
- delayed milestone execution reduces the customer's payment; a contractual penalty is imposed for the delay;
- the contractor incurs the activity execution cost connected with, *inter alia*, use of resources, materials, transport etc.;
- the expenditure is expensed at the time of planned commencement of activity execution as per the baseline schedule;
- all of the contractor's expenses are attributable to project activities.

Table 1 lists the symbols used while defining NPV optimisation model for a project settled in milestones.

Table 1
Symbols

| | | |
|---|---|---|
| $G(V, E)$ | – | acyclic directed graph depicting the project in the AON (Acivity On Node) representation; |
| $V$ | – | set of vertices (nodes) representing project activities; |
| $E$ | – | set of edges (arcs) representing ordering relations between project activities; |
| $n$ | – | number of project activities; |
| $i$ | – | index of an activity, $i$= 0, 1,…, $n$, $n$+1; activities 0 and $n$+1 represent the initial and final vertices, respectively, of the graph $G(V, E)$; |
| $d_i$ | – | duration of activity $i$; |
| $ST_i$ | – | start time for activity $i$ as per the current schedule; |
| $FT_i$ | – | finish time for activity $i$ as per the current schedule ($FT_i$ = $ST_i + d_i$); |
| $K$ | – | number of types of renewable resources; |
| $k$ | – | index of a resource type, $k = 1, \ldots, K$; |
| $a_k$ | – | number of available resources of type $k$ at any time during project execution; |
| $r_{ik}$ | – | number of type $k$ resources used in the execution of activity $i$; |
| $A(t)$ | – | set of activities being executed in the time interval $[t-1, t]$; |
| $M$ | – | number of project milestones; |
| $m$ | – | index of a project milestone, $m = 1, \ldots, M$; |
| $\delta_m$ | – | contractual completion time for milestone $m$; |
| $\Delta_m$ | – | completion time for milestone $m$ as per the current schedule; |
| $J_m$ | – | set of activities to be executed in milestone $m$; |
| $NCF_i$ | – | Negative Cash Flows, the contractor's expenditure on the execution of activity $i$, incurred at the activity start time; |
| $P_m$ | – | the customer's contractual payment for the execution of milestone $m$; |
| $C_m$ | – | contractual unit cost of delay in the execution of milestone $m$; |
| $PCF_m$ | – | Positive Cash Flows, the customer's payment for the execution of milestone $m$ determined for the current schedule, with cost of execution delay, if any, included; |
| $\alpha$ | – | discount rate; |
| $E_R$ | – | set of additional edges, i.e. pairs $(i,\ j)$ of activities with no ordering relations between them in the original activity netwerk (graph) $G(V, E)$, but with resource flows: $f(i, j, k) > 0$; |
| $E_U$ | – | set uf unavoidable arcs; |
| $f(i, j, k)$ | – | for a given resource type $k$, number of resources transferred from finishing activity $i$ to starting activity $j$. |

The NPV maximisation model with cash flows defined for activities and milestones may be described with following formulae (1)–(5).

Maximise

$$F = \sum_{i=1}^{n} (NCF_i \cdot e^{-\alpha \cdot ST_i}) + \sum_{m=1}^{M} (PCF_m \cdot e^{-\alpha \cdot \Delta_m}), \quad (1)$$

with the following constraints:

$$\forall (i, j) \in E \ : \ ST_i + d_i \leq ST_j, \quad (2)$$

$$\forall t \ \forall k : \sum_{i \in A(t)} r_{ik} \leq a_k \quad (3)$$

and with the following milestones:

$$\Delta_m = \max_{i \in J_m} (FT_i), \quad (4)$$

$$PCF_m = P_m - C_m \cdot \max(\Delta_m - \delta_m, 0). \quad (5)$$

The objective of scheduling is to determine the vector of activity start times $ST_i$ maximising the value of the objective function $F$ (see formula (1)). Ordering relations of the finish-start without zero-lag type occur between activities (see formula (2)). Activities are performed with use of limited renewable resources, whose constant availability is $a_k$ $(k = 1, \ldots, K)$ at any time $t$ (see formula (3)).

The schedule looked for should take into consideration project milestones (see formulae (4), (5)). If milestone execution is delayed, the schedule is still executable, but the customer's payments are reduced (see formula (5)), which in turn reduces aggregate DCF (see formula (1)). Milestone execution ahead of the schedule is to the contractor's benefit, owing to the customer's earlier payment and the resulting increase in the payment DCF.

The model with project settlement in milestones, developed herein, is favourable to the contractor, who thus receives, from the customer, funds which the contractor may use to finance its operations (activity execution, purchase of materials etc.) before work completion. From the customer's perspective, early payments are not advisable, but project settlement in milestones enables the customer to monitor project execution progress and the contractual penalty system urges the contractor to timely execute the project.

## 3. Iterative right-shift algorithms

To-date research into project scheduling with NPV maximisation has not covered the problem with project settlement in milestones in the form discussed herein. While milestone-related cash flows have been considered for the MMRCPSP problem [15, 16], the optimisation models used therein differ from the one presented in this paper.

In the problem under discussion, cash outflows occur at activity commencement times. Therefore, it is advisable to start activity execution as late as possible, thus reducing DCF for the contractor's expenses. On the other hand, the earlier a milestone is completed, the larger is the value of the objective function $F$ (the customer's earlier payments mean higher NPV). Thus it is beneficial to delay (right-shift in the schedule) those activities whose delay does not affect milestone execution times.

The research papers discuss various procedures (review thereof is included in [19]) for solution improvement by activity shifts. However, those procedures do not apply to a problem with predefined milestones. They are applied to RCPSPDCF models, in which activities are ascribed outflows and/or inflows [3, 19–22]. For identifying a solution dedicated to the problem discussed, the authors propose using iterative right-shift algorithms. The first one, the **RS1** algorithm, is presented in Fig. 1 [4].

The operation of the **RS1** algorithm starts with the creation of schedule $S$ in which activity execution start times are determined, taking into consideration order and resource constraints, with the optimisation of the objective function $F$, without right shifts of activities. The RCPSP problem, being a generalisation of the Job Shop problem, is NP-hard [23]; for this reason, for large projects, the solution $S$ is generated with use of approximate algorithms, such as **SA** (Simulated Annealing) metaheuristics, genetic algorithms etc. For a review of the algorithms used and comparison of their effectiveness, we refer the reader to papers [24–26]. In this paper, the authors use simulated annealing metaheuristics [27], whose effectiveness has been confirmed by testing thereof for the RCPSP problem with, *inter alia*, minimisation criterion for project execution time [24, 25, 28]. Herein, the function $F$ is used as the objective function during the search for a schedule S with **SA** metaheuristics.

```
Find schedule S;
Find resource allocation for schedule S;
S* := S;
Create empty list TL;
Repeat
        F* := F(S*);
 F' := F*;
        For i := 1 To n
        Begin
                If(i is not in list TL) Then
                Begin
                        S' := shift(S*, i);
                        If (F(S') > F') Then
                        Begin
                                i* := i;
                                F' := F(S');
                        End
                        Else
                                Add activity i to
list TL;
                End
 End
        If (F* < F') Then
                S* := shift(S*, i*);
Until (F* < F')
Return S*
```

Here: $S$ is the schedule without right shifts, $TL$ is the list of activities whose shift does not increase the value of objective function $F$, $F(S)$ is the value of objective function $F$ for schedule $S$, $S^*$ is the schedule best at the time, $S'$ is the currently analysed schedule, $F'$ is, for a given iteration, the highest of the values of objective function $F$ for schedules with various right shifts, $i$ is the number of currently analysed activity, $i^*$ is the number of the activity whose right shift generates the largest increment in the value of objective function $F$ in a given iteration and *shift* ($S^*$, $i$) is the method creating a schedule by a unit right shift of activity $i$ in schedule $S^*$.

Fig. 1. **RS1** algorithm

The schedule $S$ having been found, allocation of resources to activities is performed. Resource allocation algorithms are discussed in the next section. Predefined resource allocation renders right shifts of activities easier [4]. The problem of resource constraints is thus solved. Consequences to the schedule of a delay in activity starting admit unique determination.

At the next step of the algorithm operation, the schedule $S^*$ with resource allocation is improved. In consecutive

iterations, unit right shifts of individual activities are tested and such rearrangement of the schedule is performed which maximises the objective function $F$. The procedure stops at the iteration in which no right shift is identified which would increase the value of objective function $F$.

It should be noted that during initial iterations of the **RS1** algorithm, the right shifts performed introduce larger rearrangements of the schedule (as they simultaneously delay starting times of numerous other activities). Therefore, the right shift of a given activity group can also force the right shift of such other activities whose delay brings about an adverse effect. For this reason, the authors propose the following **RS2** procedure presented in Fig. 2.

```
Find schedule S;
Find resource allocation RA for schedule S;
Create list SL;
S* := S;
For (for each consecutive activity i in list SL)
Begin
 F* := F(S*);
 Repeat
                S' := shift (S*, i);
                If (F(S') > F*) Then
                Begin
                        S* := S';
                        F* := F(S');
                End
 Until (F* < F(S'))
End
Return S*
```

Here: $SL$ – list of activities arranged by the decreasing finishing times in schedule $S$, for the other symbols: see Fig. 1.

Fig. 2. **RS2** algorithm

In the **RS2** algorithms, shifts are performed step by step. In consecutive iterations, right shifts are analysed of activities arranged according to decreasing start times in schedule $S$. An activity is shifted right by 1, until its start so delayed stops increasing the value of $F$.

## 4. Resource allocation

For a fixed nominal schedule (that is specified activity start times $ST_1, \ldots, ST_n$), numerous resource allocations are possible for which **RS1** and **RS2** generate different solutions involving right shifts.

The resource allocation problem for the RCPSP problem is strongly NP-hard for just one resource type [29]. To model the problem, a resource flow network is used including the edges of the original activity network $G(V, E)$ and the set $E_R$ of additional edges. A resource flow network is composed of all pairs $(i, j)$ of vertices (activities) for which non-zero resource flow occurs: $f(i, j, k) > 0$. The aggregate resources of a given type entering the vertex equals the aggregate resources of the type exiting the vertex and is denoted by $r_{ik}$ (for each resource type $k$) [30]. The aggregate of all resources of a given type exiting the initial activity 0 and the aggregate of all resources of the type entering the final activity $n + 1$ are both equal to $a_k$ (for each resource type $k$).

The resource allocation problem is analysed for proactive, robust scheduling, where among the objectives, there is minimisation of the number of additional edges [30, 31], maximisation of aggregate flows between individual activities, minimisation of the effect of potential disruptions [30] etc. For a review of resource allocation algorithms, the reader is referred to papers [30, 32]. In this paper, the procedures are described used in computational experiments.

Each additional arc (edge) in $E_R$ means a new ordering constraint, which reduces schedule robustness. Intuitively, the same is true for the problem analysed: the fewer the number of additional, non-technological ordering constraints, the more room for activity shifts. Accordingly, it seems justified to use known procedures of robust resource allocation to create a resource flow network included in the iterative right-shift algorithm.

In the simplest allocation procedure, **BasicChaining**, activities are allocated to the first free chains connected with consecutive resources. Executable resource flow networks are created, without consideration of optimising criteria, frequently with an excessive cardinality of $E_R$. Iterative Sampling Heuristic (**ISH**) [31] is a procedure generating fewer additional edges than **BasicChaining** does. Activities with a demand for a given resource larger than 1 are allocated with a view to maximising the number of chains in common with the most recent activities in available chains. The **ISH** procedure ignores the original network (graph) $G(V, E)$. In the **ISH**$^2$ algorithm, each analysed activity $i = 1, \ldots, n$ is first allocated to chains in which the last activity is the direct predecessor of activity $i$. Another algorithm, **ISH-UA** [33], operates similarly to **ISH**$^2$, with the procedure of identifying unavoidable arcs launched first [30]. A given activity is first allocated to chains in which the last activity is the direct predecessor of the activity being allocated or is connected to it with an unavoidable arc.

The authors also propose the **RALS** (Resource Allocation with Local Search) algorithm [33], presented in Fig. 3.

The operation of the **RALS** algorithm starts with the identification of unavoidable arcs, which are in each iteration added to the set $E_R$. Then the *LRA* list is created defining the order of activities during allocation. Prior to the launch of the **RALS** algorithm, the activities are arranged in the *LRA* list in the increasing order of their respective start times in schedule $S$. Activities sharing the same start time $ST_i$ are arranged in the decreasing order of their respective demand for resources. The improvement of resource allocation consists in the rearrangement of activities in the *LRA* list at times at which more than one activity start. Groups are created of activities with the same start time. Local search of solutions proceeds as follows: a group of activities subject to shift is chosen at random, with the probability of choosing a group proportional to its size, which means that a group of a larger number of activities is, on average, relocated more often. Then a relocation is performed resulting in a rearrangement

```
Identify set of unavoidable arcs E_UA;
Create list LRA of activity sorted in the increasing order of start times in S; if start times are equal, sort in the decreasing order of aggregate demand for resources;
Identify groups of activities with the same start time ST;
iteration := 0; F_RA* := F(S);
Repeat
  iteration := iteration + 1;
  Clear resource flow network f and set E_R;
  Add arcs from set E_UA to set E_R;
  Select at random an activity group from list LRA for movement;
  For selected group, perform movement rearranging this group in activity list LRA, thus creating list LRA';
  For (each consecutive activity i in list LRA')
  Begin
    For(each resource type k for which r_ik > 0)
    Begin
      Clear set PA;
      To PA, add activities j which take final positions in available resource chains at time ST_i and are connected to activity i with arc, i.e. (j, i) is in (E ∪ E_R);
      While (aggregate available resources in PA < r_ik) Do
                Begin
        From among activities outside PA which take final positions in available resource chains at time ST_i, choose at random activity j and add it to set PA;
                End
      Allocate resources to activity i by creating relevant resource flows f(j,i,k) for activities in PA;
                End
  End
  If (F_RA(f) > F_RA*) Then
  Begin
                F_RA* := F_RA(f);
                f* := f;
                LRA := LRA';
  End
Until (iteration < iterMax))
Return resource flow network f*
```

Here: $F_{RA}(f)$ – value of assessment function for resource flow network $f$, $F_{RA}$* – the largest current value of assessment function, $f$* – the best current resource flow network, $f$ – currently analysed resource flow network, *LRA'* – list of activities defining the order of allocation for which resource flow network $f$* is generated, *LRA'* – currently analysed list of activities defining the order of allocation, *PA* – set of activities used for construction resource flows for actual activity, *iterMax* – number of iteration in which resource allocation is generated, *iteration* – number of current iteration.

Fig. 3. **RALS** algorithm

of activities within the chosen group, and thus in the *LRA* list. Rearrangements within a group are of three types: Swap, Insert and Random activity order [33]. Based on the order of activities in the *PA* list, resources are allocated to the current activity $i$. After the rearrangement, resource allocation is generated for the current *LRA* list, which is now compared with the then best solution. Allocation is assessed against two criteria: maximisation of the *flex* metric [34] (this criterion is equivalent to the minimisation of the number of additional arcs) and maximisation of the value of objective function $F$ determined for the schedule with shifts found with the **RS2** algorithm. The order of activities in the *LRA* list for the better of these two resource allocations is stored in the memory to be modified in the next iteration.

Resource allocation resolves itself into determining, at consecutive times $t = ST_i$, resource flows from all activities which have freed the currently available resources to activity $i$. To empty set *PA*, the activities are added which are connected to activity $i$ with an edge and which are final vertices of resource chains at time $ST_i$. If the aggregate resources freed by the activities in *PA* are lower than the demand for the resources from activity $i$, than the missing resources and the related activities are identified. From among the activities which are final vertices in available resource chains at time $ST_i$ and are not in *PA*, activities are chosen at random and added to *PA* until the aggregate resources freed by the activities in *PA* are at least equal to the demand for that resource type from activity $i$. The next step consists in resource allocation to activity $i$ by way of creating appropriate resource flows $f(j,\ i,\ k)$ between activities, with activity $j$ in *PA*. If edge $(j,\ i)$ is not in $E \cup E_R$, it is added to the set of additional arcs $E_R$ and taken into consideration in the allocation of other resource types.

## 5. Illustrative example

The exemplifying project [5] consists of eight activities performed with availability of a single resource type set at 10. Three settlement milestones are defined. For the purposes of NPV computing, the discount rate of $\alpha = 0.05$. Figure 4 presents the AON activity network for the project with all parameters of the optimisation model analysed.

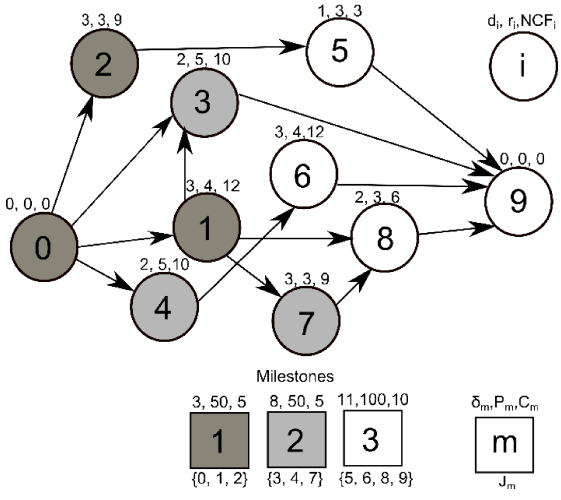


Fig. 4. Activity network with milestones

In the authors' method, an active schedule (without right shifts of activities) is created with use of a serial SGS. This schedule is then improved by right shifts of activities. A sample schedule $S$ without right shifts, with the 67.67 value of objective function $F$ is shown in Fig. 5. It can be generated, for instance, for the activity list $\{1, 2, 3, 4, 7, 6, 5, 8\}$. Activity start times in solution $S$ are: $ST_1 = 0$, $ST_2 = 0$, $ST_3 = 3$, $ST_4 = 3$, $ST_5 = 5$, $ST_6 = 5$, $ST_7 = 5$, $ST_8 = 8$, $ST_9 = 10$. All milestones are executed as scheduled ($\Delta_1 = 3$, $\Delta_2 = 8$, $\Delta_3 = 10$ against contractual deadlines $\delta_1 = 3$, $\delta_2 = 8$, $\delta_3 = 11$).

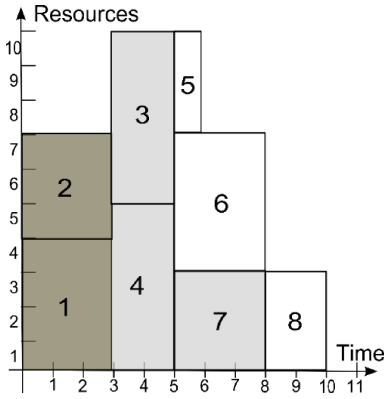


Fig. 5. Schedule $S$ without right shifts; $F = 67.67$

The value of objective function $F$ can be increased by right shifts of those activities in schedule $S$ whose delay does not affect milestone completion times. Without resource allocation to activities during the right-shift procedure it is difficult to include resource constraints in the solution. Depending on the resource allocation performed, the proposed **RS1** and **RS2** algorithms generate different schedules $S'$. Resource allocations are preferred with the lowest number of additional arcs. The additional ordering constraints resulting from the resource allocation chosen may reduce the number of activities whose shifts increase project NPV [4]. It should be noted, however, that some additional arcs would not adversely affect the quality of solutions generated by the right-shift algorithms **RS1** and **RS2**.

For schedule $S$, resources are transferred between activities at times $t = 3$, $t = 5$ and $t = 8$. There are no unavoidable arcs.

At time $t = 8$, additional arcs (6, 8) and (5, 8) may appear. For the problem and right-shift algorithms analysed, such resource allocation is preferred in which these arcs do not appear, that is when all three resources required for the execution of activity 8 are transferred from activity 7 (the precedence relation occurs between activities 7 and 8 in the original AON network). For such resource allocation, the **RS1** and **RS2** algorithms will generate a solution in which activities 5 and 6 are right-shifted by 4 and 2, respectively. With the allocation in which additional arc (5, 8) occurs, activity 5 may be right-shifted by 2 only. Now if additional arc (6, 8) occurred, the right-shift of activity 6 would reduce the project NPV as the completion of milestone 3 would then be delayed.

At time $t = 5$, any of additional arcs (3, 5), (3, 7), (3, 8), (4, 5) and (4, 7) may appear. As a result of allocation, at least three of them will appear, but none of them unavoidable. It should be noted that, irrespective of the allocation chosen, a right-shift of activity 3 would reduce the project NPV. On the other hand, a right-shift of activity 4 may increase the value of $F$. Therefore, in the course of allocation, such transfer of resources between activities is advisable which does not lead to the appearance of additional arcs which would limit the feasibility of shifting activity 4.

At time $t = 3$, any of additional arcs (1, 4), (2, 3) and (2, 4) may appear. As a result of allocation, at least one of them will appear, but none of them is unavoidable. Resource allocation at $t = 3$ has no effect on the quality of the generated solution with right shifts, because, in the exemplifying project, right shifts of activities 1 or 2 reduce the project NPV as a result of a delayed completion of milestone 1.

Figure 6a presents schedule $S$ with a sample resource allocation, with the minimum number of additional arcs. The related resource flow network is presented in Fig. 6b (with additional arcs drawn as broken arrows). Figure 6c sets forth the schedule with right shifts generated by **RS1** or **RS2**.

The solution with right shifts presented in Fig. 6c is not optimal. The value of objective function $F$ is 68.98 and is by 1.31 larger than the value for schedule $S$ in Fig. 2, owing to the right shifts of activities 5 and 6. However, the resource allocation performed brought about the appearance of arc (4, 7), whose presence renders the right shift of activity 4 adverse, as it reduces the project NPV.

Schedule $S$ with a sample resource allocation which would be optimal for the problem considered is shown in Fig. 7a. The related resource flow network is presented in Fig. 7b, while Fig. 7c sets forth the related schedule with right shifts.

For the schedule presented in Fig. 7c, the value of objective function $F$ is 69.80. The solution can be found by both **RS1** and **RS2** algorithm. The **RS1** procedure performs, in sequence, the following unit right shifts: activity 4 (objective function $F$ increases from 67.67 to 68.66, start times of activities 5 and 6 are delayed by 1, too); activity 4 ($F$ increases from 68.66 to 69.60, start times of activities 5 and 6 are delayed by 1, too); activity 5 ($F$ increases from 69.60 to 69.70); and, finally, activity 5 ($F$ increases from 69.70 to 69.80). With use of the **RS2** procedure, right shifts are performed in the order of activity finish times in schedule $S$. The activities analysed are, in sequence: activity 8 (the shift reduces $F$); activity 7 (the shift reduces $F$); activity 6 (the two-unit shift increases $F$ from 67.67 to 68.56); activity 5 (the four-unit shift increases $F$ from 68.56 to 68.98); activity 4 (the two-unit shift increases $F$ from 68.98 to 69.80); activity 3 (the shift reduces $F$); activity 2 (the shift reduces $F$); and, finally, activity 1 (the shift reduces $F$).

The schedule in Figs. 7a,c includes five additional arcs, more than the schedule in Figs. 6a,c (three additional arcs); despite this, the **RS1** or **RS2** procedure generates a solution with a higher NPV.

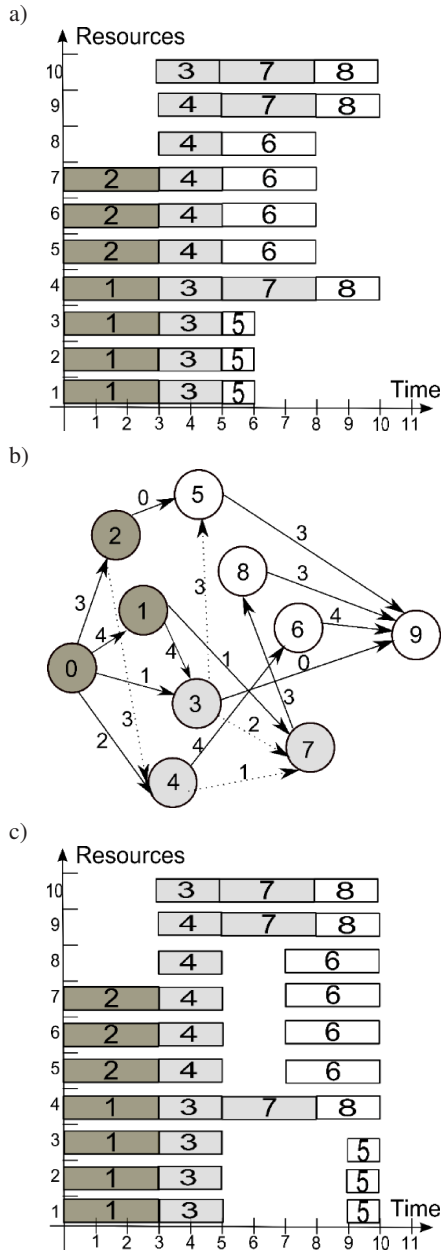


Fig. 6. a) Schedule $S$ with a sample feasible resource allocation with the minimum number of additional arcs, b) resource flow network for resource allocation of Fig. 6a, c) solution with right shifts for schedule with resource allocation of Fig. 6a

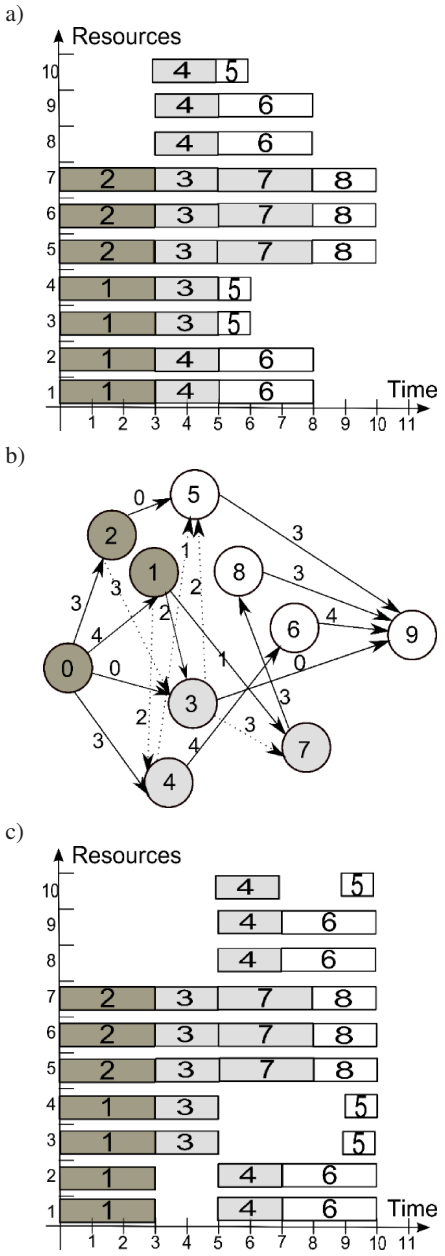


Fig. 7. a) Schedule $S$ with resource allocation for which **RS1** or **RS2** generates the best solution with right shifts, b) resource flow network for resource allocation of Fig. 7a, c) the best solution, generated by **RS1** or **RS2** for schedule with resource allocation of Fig. 7a

The example analysed proves that for the same original schedule, but different resource allocations, right-shift schedules can be generated with different values of the objective function $F$. It is reasonable to examine known resource allocation algorithms for their effectiveness in solving the problem considered. These algorithms search for allocations which minimise the quantity of the resources used. However, such allocations may not prove optimal for the problem considered. It might prove advisable to use the proposed algorithm of resource allocation with local search with a view to maximising the objective function $F$.

## 6. Computational experiments

Computational experiments were run on a computer equipped with an Intel Core I7, 3.0 GHz processor, 8 GB RAM, with use of a program developed in the C# language and in the Visual Studio.NET environment. 960 test instances from J30 (30-activity projects) and J90 (90-activity projects) from the PSPLIB library [18] were used, with additionally defined four milestones, generated by the **LOSM** procedure [33]. In each test instance, the following parameter values were assumed for financial settlements [4]: $\alpha = 0.01$, $P_1 = 40$, $C_1 = 1$,

$P_2 = 40$, $C_2 = 1$, $P_3 = 40$, $C_3 = 1$, $P_4 = 80$ and $C_4 = 2$. Costs $NCF_i$ were determined as proportional to demand for resources from, and duration of, activity $i$, assuming that aggregate costs of all activities are 100.

The experiments have been designed to verify the effectiveness of the proposed right-shift procedures **RS1**, **RS2** and the effect of the resource allocation algorithm used on the solutions generated. The experiments and analysis thereof do not cover the selection of parameters for the **SA** algorithm generating a schedule without right shifts. The following settings of the **SA** algorithm have been used [4]. Coding: activity list, forward scheduling; decoding procedure: serial SGS; initial solution and cooling scheme parameters determined in the tuning phase; number of solutions analysed in the tuning phase: 200; number of examined solutions: 5000; movement: Insert, cooling scheme: logarithmic.

All improvement procedures, with various parameters, are tested on the same schedules determined with use of the simulated annealing metaheuristics. The average value of the objective function $F$ for the schedule without right shifts, and for the J30 and J90 sets is 53.2017 and 32.7414, respectively. The use of right-shift procedures increases the project NPV for each test instance. Tables 2 and 3 set forth the results of the computational experiments run.

The choice of parameters primarily refers to a shift procedure (**RS1** or **RS2**) and resource allocation procedure (**Basic Chaining**, **ISH**, **ISH**$^2$, **ISH-UA** or **RALS**). For the search of solution space under the **RALS** algorithm, the movements Insert, Swap and Random activity order are used. The number of iterations (movements performed with the analysis of allocation obtained) is 100 or 1,000. In the course of algorithm operation, the quantity optimised is the *flex* metric or the objective function $F$ determined for a schedule with right shifts generated with the **RS2** algorithm.

For 30-activity projects (from the J30 set), all solutions with resource allocation determined with use of the **RALS** algorithm are, after 1,000 iterations, identical to the best schedule found. For 90-activity projects (from the J90 set), the **RS2** algorithm proves best, with resource allocation generated by the **RALS** procedure after 1,000 iterations, with use of the movement Random activity order and the maximisation of $F$ as the assessment function.

The right-shift procedure selected (**RS1** or **RS2**) has no material effect on the value of objective function $F$ for the schedules generated. For just few test instances, the **RS2** algorithm generates solutions slightly better than those generated by **RS1**. The solutions determined by the **RS2** procedure are in each case better or at least identical to those generated by the **RS1** procedure.

The resource allocation algorithm used has a larger effect on the NPV of schedules generated. Among simple procedures (without local search), the best solution was obtained for resource allocations determined with use of the **ISH-UA** algorithm. The **RALS** algorithm proved most effective, but also most time consuming. The best search technique is Random activity order. The research conducted indicates that the optimisation of the objective function $F$ leads to better resource allocations than the optimisation of the *flex* metric. For procedures other than **RALS**, the following relations can be observed: the higher the value of *flex* (the lower the number of additional arcs), the higher the value of the objective function $F$ for a schedule generated by **RS1** or **RS2**. On the other hand, in the event of the **RALS** procedure, in the assessment of resource allocation, the minimisation of the number of additional arcs (equivalent to the maximisation of *flex*) proves less effective than the maximisation of $F$.

Table 2
Results of computational experiments for test projects from the J30 set

| Parameters | $t_{av}$ [s] | *flex* | RS1 | | RS2 | |
|---|---|---|---|---|---|---|
| | | | $F_{av}$ | $\#F_{\max}$ | $F_{av}$ | $\#F_{\max}$ |
| **Basic Chaining** | 0.0019 | 0.302 | 53.7721 | 46 | 53.7805 | 56 |
| **ISH** | 0.0014 | 0.311 | 53.8393 | 115 | 53.8395 | 117 |
| **ISH2** | 0.0014 | 0.339 | 53.9562 | 267 | 53.9572 | 269 |
| **ISH-UA** | 0.0017 | 0.345 | 53.9924 | 376 | 53.9926 | 378 |
| **RALS**: i0, r0, m0 | 0.1109 | 0.355 | 54.0056 | 433 | 54.0057 | 434 |
| **RALS**: i0, r0, m1 | 0.1337 | 0.346 | 54.0125 | 463 | 54.0125 | 464 |
| **RALS**: i0, r1, m0 | 0.1105 | 0.355 | 54.0127 | 467 | 54.0127 | 467 |
| **RALS**: i0, r1, m1 | 0.1336 | 0.346 | 54.0133 | 470 | 54.0133 | 470 |
| **RALS**: i0, r2, m0 | 0.1107 | 0.356 | 54.0138 | 473 | 54.0138 | 473 |
| **RALS**: i0, r2, m1 | 0.1340 | 0.346 | 54.0154 | 480 | 54.0154 | 480 |
| **RALS**: i1, r0, m0 | 1.0927 | 0.355 | 54.0154 | 480 | 54.0154 | 480 |
| **RALS**: i1, r0, m1 | 1.3256 | 0.346 | 54.0154 | 480 | 54.0154 | 480 |
| **RALS**: i1, r1, m0 | 1.0925 | 0.355 | 54.0154 | 480 | 54.0154 | 480 |
| **RALS**: i1, r1, m1 | 1.3235 | 0.346 | 54.0154 | 480 | 54.0154 | 480 |
| **RALS**: i1, r2, m0 | 1.0925 | 0.357 | 54.0154 | 480 | 54.0154 | 480 |
| **RALS**: i1, r2, m1 | 1.3241 | 0.346 | 54.0154 | 480 | 54.0154 | 480 |

Here: $t_{av}$ – average computation time in the phase of resource allocation construction and activity shifts, $F_{av}$ – average value of the objective function $F$, $\#F_{\max}$ – number of solutions identical to the best ones identified by all of the algorithms developed (from among 480 test instances), **RALS** parameters: i0 – 100 iterations, i1 – 1000 iterations, r0 – movement Insert, r1 – movement Swap, r2 – movement Random activity order, m0 – maximising *flex*, m1 – maximising $F$ for a solution generated by **RS2**.

Table 3
Results of computational experiments for test projects from the J90 set

| Parameters | $t_{av}$ [s] | *flex* | RS1 | | RS2 | |
|---|---|---|---|---|---|---|
| | | | $F_{av}$ | $\#F_{\max}$ | $F_{av}$ | $\#F_{\max}$ |
| **Basic Chaining** | 0.0019 | 0.423 | 33.2227 | 0 | 33.2339 | 0 |
| **ISH** | 0.0014 | 0.427 | 33.3528 | 0 | 33.3544 | 0 |
| **ISH2** | 0.0014 | 0.444 | 33.5599 | 18 | 33.5644 | 22 |
| **ISH-UA** | 0.0017 | 0.450 | 33.6351 | 52 | 33.6359 | 52 |
| **RALS**: i0, r0, m0 | 0.1109 | 0.462 | 33.6983 | 153 | 33.6987 | 155 |
| **RALS**: i0, r0, m1 | 0.1337 | 0.449 | 33.7288 | 210 | 33.7289 | 211 |
| **RALS**: i0, r1, m0 | 0.1105 | 0.461 | 33.7343 | 222 | 33.7347 | 222 |
| **RALS**: i0, r1, m1 | 0.1336 | 0.449 | 33.7468 | 235 | 33.7468 | 235 |
| **RALS**: i0, r2, m0 | 0.1107 | 0.461 | 33.7579 | 253 | 33.7582 | 253 |
| **RALS**: i0, r2, m1 | 0.1340 | 0.449 | 33.7764 | 300 | 33.7768 | 300 |
| **RALS**: i1, r0, m0 | 1.0927 | 0.464 | 33.7788 | 310 | 33.7788 | 310 |
| **RALS**: i1, r0, m1 | 1.3256 | 0.449 | 33.7833 | 318 | 33.7835 | 318 |
| **RALS**: i1, r1, m0 | 1.0925 | 0.464 | 33.7840 | 319 | 33.7840 | 319 |
| **RALS**: i1, r1, m1 | 1.3235 | 0.449 | 33.7859 | 321 | 33.7859 | 321 |
| **RALS**: i1, r2, m0 | 1.0925 | 0.464 | 33.7882 | 327 | 33.7882 | 327 |
| **RALS**: i1, r2, m1 | 1.3241 | 0.450 | 33.8005 | 477 | 33.8011 | 480 |

For the meaning of symbols, see Table 2.

Increasing the number of iterations improves the quality of solutions found. Especially for 90-activity projects, increasing the number of iterations above the 1,000 threshold can increase the value of the objective function $F$.

## 7. Summary

The paper analyses the problem of DCF maximisation from the contractor's perspective. A new model is proposed for projects settled in milestones. Solution improvement algorithms are presented, using unit right shifts of activities for an assumed resource allocation, with a view to maximising the project NPV.

The results of the computational experiments run indicate that the effectiveness of shift procedures depends primarily on the assumed resource allocation. The best solutions are obtained, when resource allocation is determined with use of the authors-developed **RALS** algorithm (of local search) designed to optimise right shifts of activities.

The problem discussed is of a current interest and material from the practical perspective. The algorithms proposed are versatile; they are applicable to, *inter alia*, other models of NPV optimisation.

**Acknowledgements.** This research has been supported by Polish National Science Center research grant # N N519 645940.

REFERENCES

[1] A.H. Russell, "Cash flows in networks", *Management Science* 16, 357–373 (1970).

[2] M. Mika, G. Waligora, and J. Weglarz, "Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models", *Eur. J. Operational Research* 164 (3), 639–668 (2005).

[3] M. Vanhoucke, E. Demeulemeester, and W. Herroelen, "Maximizing the net present value of a project with linear time-dependent cash flows", *Int. J. Production Research* 39 (14), 3159–3181 (2001).

[4] M. Klimek and P. Lebkowski, "An algorithm for maximising discounted cash flow problem of project settled with stages", *Innovations in Management and Production Engineering* 1, 572–582 (2014), (in Polish).

[5] M. Klimek and P. Lebkowski, "Iterative right-shift algorithms for maximising the net present value of a project settled with stages", *Automation of Discrete Processes: Theory and Applications* 2, 123–141 (2014), (in Polish).

[6] M. Klimek and P. Lebkowski, "A two-phase algorithm for a resource constrained project scheduling problem with discounted cash flows", *Decision Making in Manufacturing and Services* 7 (1–2), 51–68 (2013).

[7] S. Hartmann and D. Briskorn, "A survey of variants and extensions of the resource-constrained project scheduling problem", *Eur. J. Operational Research* 207 (1), 1–14 (2012).

[8] W. Herroelen, B.D. Reyck, and E. Demeulemeester, "Project network models with discounted cash flows: A guided tour through recent developments", *Eur. J. Operational Research* 100, 97–121 (1997).

[9] N. Dayanand, R. Padman, "On modelling payments in projects", *J. Operational Research Society* 48, 906–918 (1997).

[10] G. Ulusoy, F. Sivrikaya-Serifoglu, and S. Sahin, "Four payment models for the multi-mode resource constrained project scheduling problem with discounted cash flows", *Annals Operations Research* 102, 237–261 (2001).

[11] N. Dayanand and R. Padman, "Project contracts and payment schedules: the client's problem", *Management Science* 47, 1654–1667 (2001).

[12] G. Ulusoy and S. Cebelli, "An equitable approach to the payment scheduling problem in project management", *Eur. J. Operational Research* 127 (2), 262–278 (2000).

[13] Z. He and Y. Xu, "Multi-mode project payment scheduling problems with bonus penalty structure", *Eur. J. Operational Research* 189 (3), 1191–1207 (2008).

[14] M. Klimek and P. Lebkowski, "Robustness of schedules for project scheduling problem with cash flow optimisation", *Bull. Pol. Ac.: Tech.* 61 (4), 1005–1015 (2013).

[15] Z. He, N. Wang, T. Jia, and Y. Xu, "Simulated annealing and tabu search for multimode project payment scheduling", *Eur. J. Operational Research* 198 (3), 688–696 (2009).

[16] Z. He, N. Wang, T. Jia, and Y. Xu, "Metaheuristics for multi-mode capital-constrained project payment scheduling", *Eur. J. Operational Research* 223 (3), 605–613 (2012).

[17] R. Kolisch, "Serial and parallel resource-constrained project scheduling methods revisited: theory and computation", *Eur. J. Operational Research* 90 (2), 320–333 (1996).

[18] R. Kolisch and A. Sprecher, "PSPLIB – a project scheduling library", *Eur. J. Operational Research* 96, 205–216 (1997).

[19] M. Vanhoucke, "A scatter search procedure for maximizing the net present value of a resource-constrained project with fixed activity cash flows", *Working Paper* 2006/417, 1–23 (2006).

[20] S.M. Baroum and J.H. Patterson, "The development of cash flow weight procedures for maximizing the net present value of a project", *J. Operations Management* 14 (3), 209-27 (1996).

[21] G. Ulusoy and L. Özdamar, "A heuristic scheduling algorithm for improving the duration and net present value of a project", *Int. J. Operations and Production Management* 15, 89–98 (1995).

[22] J.P. Pinder and A.S. Marucheck, "Using discounted cash flow heuristics to improve project net present value", *J. Operations Management* 14, 229–240 (1996).

[23] J. Blazewicz, J. Lenstra, and A. Kan, "Scheduling subject to resource constraints – classification and complexity", *Discrete Applied Mathematics* 5, 11–24 (1983).

[24] S. Hartmann and R. Kolisch, "Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem", *Eur. J. Operational Research* 127, 394–407 (2000).

[25] R. Kolisch and S. Hartmann, "Experimental investigation of heuristics for resource-constrained project scheduling: an update", *Eur. J. Operational Research* 174, 23–37 (2006).

[26] R. Kolisch and R. Padman, "An integrated survey of deterministic project scheduling", *OMEGA Int. J. Management Science* 29, 249–272 (2001).

[27] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by simulated annealing", *Science* 220, 671–680 (1983).

[28] K. Bouleimen and H. Lecocq, "A new efficient simulated annealing algorithm for the resource constrained project scheduling problem and its multiple version", *Eur. J. Operational Research* 149, 268–281 (2003).

[29] R. Leus and W. Herroelen, "Stability and resource allocation in project planning", *IIE Trans.* 36 (7), 667–682 (2004).

[30] F. Deblaere, E. Demeulemeester, W. Herroelen, and S. Van De Vonder, "Proactive resource allocation heuristics for robust project scheduling", *Research Report KBI_*0608, CD-ROM (2006).

[31] N. Policella, "Scheduling with uncertainty – a proactive approach using partial order schedules", *PhD Thesis,* University La Sapienza, Rome, 2005.

[32] M. Klimek and P. Lebkowski, "Resource allocation for robust project scheduling", *Bull. Pol. Ac.: Tech.* 59 (1), 51–55 (2011).

[33] M. Klimek, "Predictive-reactive production scheduling with resource availability constraints", *PhD Thesis*, AGH University of Science and Technology, Kraków, 2010, (in Polish).

[34] M. Aloulou and M. Portmann, "An efficient proactive reactive scheduling approach to hedge against shop floor disturbances", *1st Multidisciplinary Conf. Scheduling: Theory and Applications* 1, 337–362 (2003).