J Sched (2007) 10: 311–326
DOI 10.1007/s10951-007-0031-y

The discrete time/cost trade-off problem: extensions and heuristic procedures

Mario Vanhoucke · Dieter Debels

Published online: 7 August 2007

Abstract Time/cost trade-offs in project networks have been the subject of extensive research since the development of the critical path method (CPM) in the late 50s. Time/cost behaviour in a project activity basically describes the trade-off between the duration of the activity and its amount of non-renewable resources (e.g., money) committed to it. In the discrete version of the problem (the discrete time/cost trade-off problem), it is generally accepted that the trade-off follows a discrete non-increasing pattern, i.e., expediting an activity is possible by allocating more resources (i.e., at a larger cost) to it. However, due to its complexity, the problem has been solved for relatively small instances.

In this paper we elaborate on three extensions of the well-known discrete time/cost trade-off problem in order to cope with more realistic settings: time/switch constraints, work continuity constraints, and net present value maximization. We give an extensive literature overview of existing procedures for these problem types and discuss a new meta-heuristic approach in order to provide near-optimal heuristic solutions for the different problems. We present computational results for the problems under study by comparing the results for both exact and heuristic procedures. We demonstrate that the heuristic algorithms produce consistently good results for two versions of the discrete time/cost trade-off problem.

Keywords Project scheduling · Time/cost tradeoff · Net present value

M. Vanhoucke (✉) · D. Debels
Department of Management Information, Operations Management and Technology Policy, Faculty of Economics and Business Administration, Ghent University, Tweekerkenstraat 2, 9000 Gent, Belgium
e-mail: mario.vanhoucke@UGent.be

D. Debels
e-mail: dieter.debels@UGent.be

M. Vanhoucke
Operations and Technology Management Center, Vlerick Leuven Gent Management School, Reep 1, 9000 Gent, Belgium
e-mail: mario.vanhoucke@vlerick.be

1 Introduction

Since the development of the critical path method (CPM), time/cost trade-offs have been the subject of extensive research in project scheduling. The trade-off involves a non-increasing pattern between the duration of an activity and its amount of non-renewable resource use, i.e., expediting an activity is possible by allocating more resources (i.e., at a larger cost) to it. This well-known problem has been studied under various assumptions.

The early endeavors to incorporate time/cost trade-offs in project networks assumed that the direct activity cost functions are *linear* non-increasing functions, i.e., the activity costs are a linear function of the activity durations, which are bounded from below (crash duration) and from above (normal duration). The objective was to determine the activity durations and to schedule the activities in order to minimize the project costs, i.e., the sum of the direct activity costs and the time-dependent indirect project costs, within a specified project deadline. Solution procedures for the linear case were proposed by Kelley and Walker (1959), Fulkerson (1961), Kelley (1961), Ford and Fulkerson (1962), Siemens (1971), Goyal (1975), and Elmaghraby and Salem (1984). Several other forms of activity cost functions have been studied, such as *concave* (Falk and Horowitz 1972), *convex* (Lamberson and Hocking 1970; Kapur 1973; Siemens and Gooding 1975; Elmaghraby and Salem 1982), or even *general* continuous activity cost functions (Moder et al. 1983).

Due to its practical relevance, procedures have been developed for solving the *discrete* version of the problem. This discrete time/cost trade-off problem (further abbreviated as the *DTCTP*) occurs, when the duration of project activities is a discrete, non-increasing function of the amount of a single non-renewable resource committed to them. It involves the selection of a set of execution modes (the time-cost tuples for each activity) in order to achieve a certain objective. In the literature, the problem objective has been divided into three parts. The so-called *deadline* problem (problem $1, T|cpm, \delta_n, disc, mu|av$ following the classification scheme of Herroelen et al. 1999) aims at minimizing the total cost of the project while meeting a given deadline, whereas the *budget* problem (problem $1, T|cpm, disc, mu|C_{\max}$) involves minimizing the project duration without exceeding a given budget. A third objective is to construct the complete and efficient time/cost profile over the set of feasible project durations (problem $1, T|cpm, disc, mu|curve$). This complete curve can be found by means of a horizon-varying approach, which involves the iterative solution of the deadline problem over the feasible project durations. This problem type has been investigated by numerous researchers, such as Crowston and Thompson (1967), Crowston (1970), Robinson (1975), Billstein and Radermacher (1977), Wiest and Levy (1977), Hindelang and Muth (1979), Patterson and Harvey (1979), Bianco and Speranza (1990), Vercellis (1990), Elmaghraby and Kamburowski (1992), De et al. (1995, 1997), Demeulemeester et al. (1996, 1998), Skutella (1998), and Akkan et al. (2000a, 2000b). In the remainder of this paper, we focus on the deadline version of the discrete time/cost trade-off problem under different assumptions.

Despite the large amount of literature of the discrete time/cost trade-off problem, little has been done in extending the problem to more realistic settings or solving large-scaled problems with heuristic procedures. Skutella (1998) presents approximation algorithms for the discrete time/cost trade-off problem, while Deineko and Woeginger (2001) discuss the hardness of finding approximations for that problem type. For an excellent review, we refer the reader to De et al. (1995). In our paper, we review three extensions to the classical *discrete time/cost trade-off problem* from literature in order to meet the requirements of real-life scheduling problems, i.e., time-switch constraints, work continuity constraints, and net present value optimization. Since De et al. (1997) have proven that the DTCTP is strongly NP-hard, we present a new meta-heuristic approach for these extended DTCTP versions. In the computational results section, we test whether the heuristic solutions can offer a valuable alternative for the exact algorithms.

Time-switch constraints have been introduced by Yang and Chen (2000) as a logical extension of the analyses and achievements of Chen et al. (1997) and have only been incorporated in the discrete time/cost trade-off problem by Vanhoucke et al. (2002) and Vanhoucke (2005). In these papers the deadline version of the *discrete time/cost trade-off problem with time-switch constraints* (problem $1, T|tsc, cpm, \delta_n, disc, mu|av$) has been solved for a special type of time-switch constraints in which each activity follows one of three possible work/rest patterns: Firstly, if an activity follows a *day*-pattern it can only be executed during day time, from Monday till Friday. Secondly, an activity follows a *d&n*-pattern if it can be executed during the day or night, from Monday till Friday. Finally, a *dnw*-pattern means that the corresponding activity can be in execution every day or night, and also during the weekend. In the computational results section of the current paper, we rely on the exact procedure of Vanhoucke (2005), since it has been proven that it outperforms Vanhoucke et al. (2002).

Work continuity constraints have been defined by El-Rayes and Moselhi (1998) in order to model the timely movement of project resources and hence to maintain continuity of work. The importance of this feature has been highlighted by several authors, such as De Boer (1998) for *spatial resources*, Gong (1997) for *time dependent cost* (*TDC*), and Goto et al. (2000) for *time dependent cost resources*. In the construction projects literature, several scheduling methodologies have been proposed under different names, such as the Line of Balance (LOB) method or the Linear Scheduling Method (LSM). Harris and Ioannou (1998) give an excellent overview and integrate these methods into the so-called *repetitive scheduling method* (RSM). A practical example of work continuity constraints in the repetitive construction industry can be found in Vanhoucke (2006). The *discrete time/cost trade-off problem with work continuity constraints* (*DTCTP-wc*), to the best of our knowledge, has only been studied by Reda (1990). The author presents a linear programming formulation for a repetitive project-scheduling problem where the duration of each activity can be decreased by allocating more resources at additional direct costs. The goal is to finish the project with a prespecified target duration and at a minimum direct cost. The constraints are to maintain production rates and continuity of work. Unfortunately, no computational results have been presented.

Net present value optimization in project scheduling has been investigated under various project assumptions, and has been discussed by Herroelen et al. (1997). In these problems types, cash flows are associated with activities or (milestone) events, and the objective is to maximize the discounted value of these cash flows. Despite the growing amount of research papers about net present value maximization, the only procedure for the *discrete time/cost trade-off problem with net present value maximization* (*DTCTP-*

npv, or problem type $1, T|cpm, \delta_n, disc, mu|npv$[1]) is, to the best of our knowledge, the procedure of Erengüç et al. (1993). In this paper the authors show that the *DTCTP-npv* is a mixture of the well-known time/cost trade-off problem and the payment-scheduling problem. They have developed a Benders decomposition approach to solve the problem to optimality and have tested their procedure on 140 project networks with up to 64 activities and 30 nodes.

The remainder of the paper is as follows: The next section describes a general problem formulation for the DTCTP and its three extensions to time-switch constraints (*tsc*), work continuity constraints (*wc*), and net present value (*npv*) maximization. We show that the three extensions from literature do not change the problem fundamentally. Section 3 describes a meta-heuristic approach that is able to generate near-optimal solutions for the different problems under study. In Sect. 4 we report on computational results. Section 5 draws overall conclusions and suggestions for future research.

2 The problem formulation

In the remainder of this paper, we assume that a project is represented by an activity-on-the-arc network $G = (N, A)$ where the set of nodes, N, represents network events and the set of arcs, A, represents the activities of the project. The nodes of the network are numbered from the single start node 1 to the single end node n. Each activity has M_{ij} *modes*, represented by a tuple $(d_{ij}(k), c_{ij}(k))$ with $k = 1, \ldots, M_{ij}$. The duration $d_{ij}(k)$ of an activity $(i, j) \in A$ is a discrete, non-increasing function of the amount of a single non-renewable resource (money, $c_{ij}(k)$) allocated to it, i.e., $d_{ij}(1) < d_{ij}(2) < \cdots < d_{ij}(M_{ij})$ and $c_{ij}(1) > c_{ij}(2) > \cdots > c_{ij}(M_{ij})$. Other characteristics depend on the problem type under study and are the topic of the following subsections. A solution can be represented by a selected set of modes $m_{ij} = (d_{ij}(k), c_{ij}(k))$ (with $k \in \{1, \ldots, M_{ij}\}$) for each activity (i, j) such that the total cost $\sum_{(i,j)\in A} c_{ij}(k)$ is minimized. In the remainder of this section, we show that the general model description can easily be extended to the three aforementioned extensions: time-switch constraints, work continuity constraints and net present value optimization.

Time switch constraints: In the original DTCTP, it is assumed that an activity can start at any time after the finishing of all its predecessors. The DTCTP-*tsc* assumes that activities are forced to start in a specific time interval and are down in some specified rest interval. Without loss of generality, we have incorporated a special type of time-switch constraints in which each activity follows one of three possible work/rest patterns (*day*, *d&n* and *dnw*, see Vanhoucke 2005).

Work continuity constraints: The *DTCTP-wc* minimizes the total cost of the schedule, that consists of the sum of both the direct activity costs (resulting from the selection of a mode for each activity) and work continuity cost for each activity group $A' \subset A$. The latter cost can be minimized by minimizing the time-span between the first and last activity of the activity group A'. Indeed, the resources are needed from the start of the first activity and will only be released at the completion of the last activity of the activity group. Consequently, the starting times of all intermediate activities have no influence on the idle time of this resource and, therefore, do not influence the total work continuity cost. We show that the *DTCTP-wc* can easily, and efficiently, be solved by any algorithm for the *DTCTP* by the incorporation of two extra arcs per activity group. For the sake of clarity, we introduce the following symbols:

A'	Set of activities (activity group $A' \subset A$) that require a common set of resources subject to work continuity constraints;
c_w	Work continuity cost, i.e. the cost per time unit for the set of resources of the activity group;
es_{t_i}	Earliest possible realization time of event i ($i \in N$) based on forward calculations with all activity modes on their crash duration;
ls_{t_i}	Latest possible realization time of event i ($i \in N$) based on backward calculations (given a project deadline δ_n) with all activity modes on their crash duration.

The *DTCTP-wc* involves a trade-off between the direct cost of an activity and the total work continuity cost (i.e., the work continuity unit cost c_w times the total timespan of activity group A'). Indeed, increasing the duration of an activity of A' immediately results in a decrease of the activity's direct cost, but might lead to an increase in the total duration of the activity group and, consequently, to an increase in the total work continuity cost. This trade-off can be embedded in the network by adding two extra arcs for each activity group (referred to as arc x and arc y) with each a linear time/cost profile as shown in Fig. 1.

In Fig. 1 we refer to node i as the event denoting the start of the first activity of A', i.e., the realization time of node i equals the starting time of the first activity of A'. Node j is used to refer to the event denoting the finish of the last activity of A'. Consequently, the activities (immediate arcs)

[1]Note that we refer to the *DTCTP-npv* as problem $1, T|cpm, \delta_n, disc, mu|npv$. This is in contrast to the time-switch constraints (*tsc*), which can be considered as a variant of preemption (work periods alternated by rest periods). Therefore, we have placed the abbreviation *tsc* in the second field (the so-called β-field). This is not the case for the *DTCTP-npv* where the objective (the so-called γ-field) is to maximize the net present value. Hence, we placed the abbreviation *npv* in the last field of the problem description.

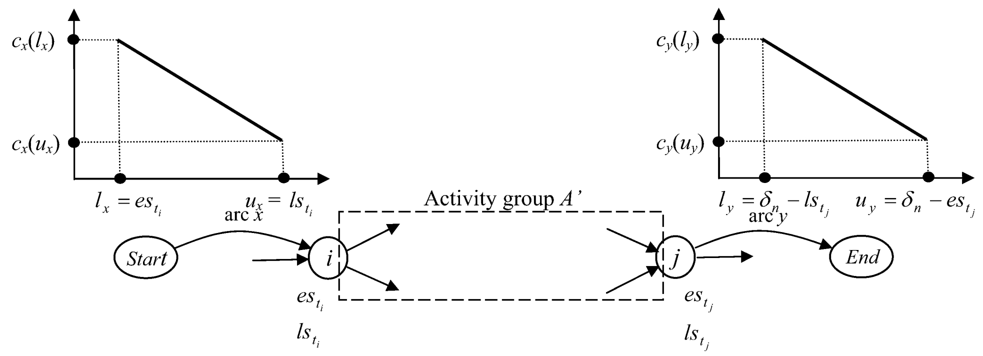

Fig. 1 Two additional arcs for solving the *DTCTP-wc*

emanating from node i are all part of A', while all incoming arcs of node j also belong to A'. This is always possible due to the introduction of two extra nodes and some dummy arcs in the network. The minimal total duration of the activity group A' equals $es_{t_j} - es_{t_i} = ls_{t_j} - ls_{t_i}$, while the maximal total duration of the activity group A' equals $ls_{t_j} - es_{t_i}$. Consequently, the minimal total work continuity cost of the project equals $c_{\min} = (es_{t_j} - es_{t_i}) * c_w = (ls_{t_j} - ls_{t_i}) * c_w$, and the maximal total work continuity cost of the project with a given deadline δ_n equals $c_{\max} = (ls_{t_j} - es_{t_i}) * c_w$. All intermediate points between these two extremes follow a linear behaviour and lie between $c_{\min}$ and $c_{\max}$.

The duration of arc x can vary between the crash duration $l_x = es_{t_i}$ and the normal duration $u_x = ls_{t_i}$, with the corresponding costs of $c_x(l_x) = \frac{c_{\max}}{2}$ and $c_x(u_x) = \frac{c_{\max}}{2} - (ls_{t_i} - es_{t_i}) * c_w$. Arc y can vary between $l_y = \delta_n - ls_{t_j}$ and $u_y = \delta_n - es_{t_j}$ with similar costs $c_y(l_y) = \frac{c_{\max}}{2}$ and $c_y(u_y) = \frac{c_{\max}}{2} - (ls_{t_j} - es_{t_j}) * c_w$. In doing so, we assure that the total work continuity cost will always[2] lie between $c_{\min}$ and $c_{\max}$, and follows a linear behaviour with slope c_w. Since the introduction of these arcs implicitly takes the trade-off between total activity cost and work continuity into account, this problem can be solved by any algorithm for the *DTCTP*.

Net present value optimization: The deadline version of the discrete time/cost trade-off problem with net present value maximization involves the scheduling of project activities in order to maximize the net present value of the project subject to precedence relations. To that purpose, positive cash flows are associated to the project events (nodes), while costs are associated to the different activities (arcs) by means of their discrete time/cost profile. We use $C_j^+ \geq 0$ to denote the positive payment received at the realization of event j. The cash outflow $c_{ij}(k)$ of each activity (i, j) depends on the selected mode k. We assume that, without loss of generality, this cash outflow occurs at the completion of each activity. This is a reasonable assumption, since it is always possible to calculate a terminal value of each activity's cash flow upon completion by compounding the associated cash flow to the end of the activity as follows: $c_{ij}(k) = \sum_{t=1}^{d_{ij}(k)} c_{ij}^t(k) e^{\alpha(d_{ij}(k)-t)}$, where α represents the discount rate, $d_{ij}(k)$ the duration of activity (i, j) at mode k and $c_{ij}^t(k)$ the value of the known and deterministic cash outflow (i.e., the cost) of activity (i, j) at mode k in period t of its execution. Consequently, the net cash flow of each event j equals $cf_j(k) = C_j^+ - \sum_{(i,j)\in\overline{A}_j} c_{ij}(k)$, with $\overline{A}_j$ the set of all incoming arcs of event j.

[2]As an example: $c_x(l_x) + c_y(u_y) = \frac{c_{\max}}{2} + \frac{c_{\max}}{2} - (ls_{t_j} - es_{t_j}) * c_w = c_{\max} - (ls_{t_j} - es_{t_j}) * c_w = (ls_{t_j} - es_{t_i}) * c_w - (ls_{t_j} - es_{t_j}) * c_w = (es_{t_j} - es_{t_i}) * c_w = c_{\min}$.

3 A meta-heuristic procedure

The heuristic search procedure boils down to the consecutive and controlled selection of a mode $m_{ij} = (d_{ij}(k), c_{ij}(k))$ (with $k \in \{1, \ldots, M_{ij}\}$) for each activity (i, j) in the network. Consequently, each iteration of the heuristic search consists of the discrete time/cost problem under study with a given set of activity durations and reduces to a much simpler problem.

The *DTCTP* with a given fixed set of activity durations reduces to the basic CPM problem (problem $cpm|C_{\max}$), which can be efficiently solved by determining the earliest completion time of each activity, using the traditional forward pass critical path calculations. This is also true for the *DTCTP-wc*, since this problem type is similar to the *DTCTP*, apart for the two extra arcs per activity group. A slight adaptation is necessary for the *DTCTP-tsc* for which the traditional forward pass calculations need to take the time-switch constraints into account (problem $cpm, tsc|C_{\max}$). The *DTCTP-npv* with a given fixed activity duration reduces to the well-known max-*npv* problem (problem $cpm|npv$), which can be efficiently solved by the recursive search procedure of Vanhoucke et al. (2001). In order to use this activity-on-the-node procedure, the AoA project network needs to be considered as an AoN network: each event j is then an activity with zero duration and a cash flow $cf_j(k)$ and the arcs represent the precedence relations with time-lags $l_{ij} = d_{ij}(k)$.

3.1 The meta-heuristic search

The meta-heuristic search procedure basically consists of three steps: an initialization step, a neighborhood search and a diversification step. Both the second and third step will be repeated until a specified stop criterion is met, as shown in the pseudo-code below.

```
Procedure Heuristic_search
  Step 1: Initialization
  While (stop criterion not met)
    Step 2: Neighborhood search
    Step 3: Diversification
  End while
Return
```

In the following sections we focus on the different steps of the heuristic search procedure in detail. In order to understand the pseudo-code, we use the following definitions:

m_{ij}	Current mode of arc (i, j) in the search process, with $m_{ij} = 1, \dots, M_{ij}$;
$tabu_{ijk}$	Tabu list of arc (i, j) in mode k;
f_{ijk}	Number of times a solution has been found with arc (i, j) in mode k (frequency);
LB	Local best solution found (within one neighborhood search iteration);
CB	Current (global) best solution found.

Remark that we use the following auxiliary variables (e.g., a counter, a Boolean variable, a parameter, etc.): we use $count = (count_1)$ to denote auxiliary variables that can have arbitrary values, $bool = (bool_1, bool_2, bool_3)$ to denote Boolean auxiliary variables (true/false), and $par = (par_1, par_2, par_2, par_4)$ to denote parameters that might affect the efficiency of the algorithm. The auxiliary variables are as follows:

$count_1$	Count the number of consecutive non-improving moves;
$bool_1$	A move has been found that is better than the current best solution (true/false) after the execution of the neighborhood search (step 2) (true/false);
$bool_2$	A move has been found that is better than the current best solution (true/false);
$bool_3$	A move has been found that is better than the local best solution (true/false);
par_1	Stop criterion for neighborhood search (number of iterations);
par_2	Length of tabu list;
par_3	Threshold for the diversification part;
par_4	Stop criterion for dynamic programming (number of backtracking steps).

Step 1 initializes the different variables and constructs a start solution. The start solution consists of a network with each activity on its crash mode (denoted by $m_{ij} = 1$) always resulting in a feasible schedule, i.e., a schedule with $C_{\max} \leq \delta_n$ and a total corresponding cost R. This can be obtained by means of the procedures mentioned above (the schedule (m_{ij})-step can be $cpm|C_{\max}$, cpm, $tsc|C_{\max}$ or $cpm|npv$). Since the procedure can possibly be improved by increasing some activity durations, the mode-set serves as an input for the procedure "truncated dynamic programming". The procedure performs a truncated dynamic programming step in order to improve the schedule found. In doing so, the existing mode set m_{ij} will be transformed into the set $m'_{ij} \geq m_{ij}$, resulting in a total cost $R' \leq R$ and $C_{\max} \leq C'_{\max} \leq \delta_n$. The truncated dynamic programming step is explained in Sect. 3.3. The solution (i.e., mode set $m_{ij} = m'_{ij}$) serves as a start input for the second and third step. In the pseudo-code below, both Steps 1 and 3 are displayed in detail.

```
Procedure Heuristic_Search
  $\forall (i, j) \in A$ and $\forall k \in M_{ij}$ set $f_{ijk} = tabu_{ijk} = 0$
  Set $CB = \infty$, $bool_1 = $ true and $\forall (i, j) \in A$ set $m_{ij} = 1$
  Schedule($m_{ij}$) + Truncated Dynamic Programming( ) $\to$
  $(m'_{ij}, R')$
  Set $m_{ij} = m'_{ij}$
  While ($bool_1$)
    Step 2: Neighborhood search
    $\forall (i, j) \in A$ and $\forall k \in M_{ij} | k < m_{ij}$
      If ($f_{ijk} > par_3$) then $tabu_{ijk} = par_2$
      $m_{ij} = rand(1, \dots, m_{ij})$
  End while
Return
```

Step 2 performs an extensive neighborhood search and selects in each iteration the best adjacent solution of the current solution. We use $bool_1$ as a Boolean variable, which initially is set to the value false in the neighborhood search procedure. From the moment the neighborhood search has found a solution with a smaller total cost than the current best solution (an improvement move), the variable is set to true. After the neighborhood search, the algorithm performs a diversification step (Step 3) and repeats the while-loop if $bool_1$ is true. The algorithm stops if no improvement move has been found during Step 2. A detailed explanation of this neighborhood step is given in Sect. 3.2.

The third step—the diversification—consists of two steps. First, the diversification is based on frequency counts f_{ijk} of Step 2 such that all activity-mode combinations that have been evaluated frequently (more than par_3 times) are set tabu during a part of the next neighborhood search (parameter par_2). Secondly, a new solution will be generated, starting from the last solution found, such that the new mode selection m_{ij} is randomly selected between the crash mode and the current mode. In doing so, we assure that the new solution is feasible, i.e., $C_{\max} \leq \delta_n$.

3.2 The neighborhood search

The second step scans the whole neighborhood and selects the best adjacent solution of the current solution m_{ij}. Moreover, each move is the subject to a local improvement step by means of a truncated dynamic programming procedure. In line with the tabu-search meta-heuristic, each time a move is selected, this move is classified as tabu for a number of iterations (parameter par_2) to incorporate the short-term frequency-based memory. As mentioned in the previous section, the search also incorporates long-term frequency-based memory by counting the distribution of selected moves throughout the search process. These counts will be used to diversify the search in two ways. Firstly, these counts will be used in the diversification step in order to set the status of a number of possible moves tabu (see Step 3 of Sect. 3.1). Next, an implicit diversification has been incorporated in the evaluation process of the moves by assigning a penalty function in the evaluation function $(R' + f_{ijk})$. Consequently, a move that has already been selected a lot of times is heavily penalized and loses its attractiveness.

The neighborhood of a mode selection m_{ij} is defined as all mode assignments k for each arc (i, j) such that $k < m_{ij}$. In doing so, we evaluate all possible so-called crashing alternatives and guarantee that a new mode selection does not lead to an infeasible schedule, since $d_{ij}(k) < d_{ij}(m_{ij})$ if $k < m_{ij}$. Each iteration of the neighborhood search evaluates all possible moves from m_{ij} to k, and consequently, the maximum number of possible moves equals $\sum_{(i,j)}^{A}(M_{ij} - 1)$ and occurs if all activities have been assigned their normal duration. Each move is also subject to the truncated dynamic programming improvement step, since crashing an activity possibly provides room for other activities to increase their duration (see Sect. 3.3). The result is a set of new mode assignments m'_{ij} and a corresponding cost R'. The complete neighborhood will be scanned until par_1 consecutive iterations without any improvement have been made. The pseudo-code of the neighborhood search step is written below.

```
Procedure Neighborhood_Search
  Set $bool_1$ = false
  While ($count_1 < par_1$)
    $\forall(i, j, k)|(i, j) \in A$ and $k \in M_{ij}$: set $tabu_{i^*j^*k^*} = \max(0, tabu_{i^*j^*k^*} - 1)$
    $LB = \infty$, $bool_2 = bool_3$ = false
    $\forall(i, j) \in A$
      $\forall k \in M_{ij}|k < m_{ij}$                {neighborhood}
        Schedule($k$) + Truncated Dynamic Programming( ) $\rightarrow (m'_{ij}, R')$
        If ($R' \le CB$) then                {aspiration by objective}
          Set $LB = CB = R'$, $bool_1 = bool_2$ = true, $count_1 = 0$,
            $i^* = i$, $j^* = j$, $k^* = k$ and $\forall(i, j) \in A$: $m''_{ij} = m'_{ij}$
            and save the mode-set as current best solution
        Else if ($tabu_{ijk} = 0$ and $R' + f_{ijk} < LB$ and $bool_2$ = false) then
          Set $LB = R'$, $bool_3$ = true, $i^* = i$, $j^* = j$,
          $k^* = k$ and $\forall(i, j) \in A : m''_{ij} = m'_{ij}$
    If ($bool_2$ or $bool_3$) then
      Set $tabu_{i^*j^*k^*} = par_2$, and
      $\forall(i, j) \in A$: $m_{ij} = m''_{ij}$ and $f_{ijm''_{ij}} = f_{ijm''_{ij}} + 1$
    Set $count_1 = count_1 + 1$
  End while
Return
```

The approach taken by our heuristic procedure relies on a number of basic principles of the tabu-search meta-heuristic (Glover 1986). The tabu principle is the cornerstone of the tabu-search meta-heuristic and is implemented in our heuristic search procedure as a short-term memory to prevent cycling. The tabu restrictions can be overridden by means of two aspiration criteria: aspiration by default and aspiration by objective. The aspiration by default criterion is not explicitly mentioned in the pseudo-code and selects the least deteriorating move, when all available moves are classified tabu. The aspiration by objective criterion overrides the tabu restrictions if a move has been evaluated which leads to the current best solution, i.e., if $(R' \le CB)$. In the next section, the truncated dynamic programming procedure, which serves as a local search each time a move has been performed, will be explained into detail. Note that we also have incorporated a frequency-based long-term memory f_{ijk} in order to diversify the search in two ways as mentioned earlier.

3.3 Truncated dynamic programming as local search

Since the neighborhood space consists of evaluating all crashing possibilities of the current solution, it is likely that, after each move, one or more activities can be increased in duration without violating the project deadline δ_n. This step aims at improving the current mode set m_{ij} (with a resulting project duration $C_{\max}$ and a total cost R) to m'_{ij} (with $R' \le R$ and $C_{\max} \le C'_{\max} \le \delta_n$) and can, therefore be considered as a local search procedure applied to the current solution, as follows:

```
Procedure Truncated Dynamic Programming
  Step 1: random sequence of activities and modes (list $L$)
  Step 2: Dynamic programming ($par_4$)
```

It is tempting to improve the current mode set m_{ij} by evaluating all possible combinations to increase activity durations by means of a dynamic programming procedure. This would lead to the best possible solution given the start solution m_{ij}, but time-restrictions render this method inapplicable. We, however, opt for a truncated version of a dynamic programming that will be stopped after a prespecified number of backtracking steps. In doing so, we evaluate only a subset of possible improvement scenarios in a very fast and efficient way. In order to prevent that the truncated DP ends with similar solutions, regardless of the starting solution, we vary the sequence in which activities are increased in duration by creating a list L with a random order of activity/mode combinations. This list L contains, in a random order, all activity/mode combinations, except those for the current move (i', j') in the neighborhood search, as follows: $L = \{(i, j, k)|(i, j) \in A\backslash\{(i', j')\}$ and $k = m_{ij}, m_{ij} + 1, \ldots, M_{ij}$ and $\forall k' > k$: $(i, j, k) \to (i, j, k')\}$, where "$a \to b$" is used to denote that a comes before b in the list L. We exclude activity (i', j') from the DP, since otherwise the move that is subject to the DP can be canceled out, leading to cycles in the neighborhood search.

This list is then used as an input for the truncated DP, which increases the durations of the activities according to the order in the list. The dynamic programming procedure is truncated after a finite number of backtracking steps (more precisely, after par_4 backtracking steps) and an improved solution m'_{ij} is reported with a total cost $R' \leq R$ and $C_{\max} \leq C'_{\max} \leq \delta_n$. In our computational experience section, we show that the heuristic search procedure performs very well, even with a small value for par_4.

4 Computational results

In order to test the performance of our meta-heuristic procedure for the *DTCTP* under the different assumptions, we have coded both the exact and heuristic procedures in Visual C++ version 6.0 and run on a Toshiba personal computer with a Pentium IV 2 GHz processor under Windows XP. In order to evaluate the quality of the heuristic solutions, we compare them with exact solutions for all four problem types. The *DTCTP* and the *DTCTP-wc* instances will be solved to optimality by the procedure of Demeulemeester et al. (1998). The *DTCTP-tsc* instances will be solved by the exact procedure of Vanhoucke (2005). The exact procedure for the *DTCTP-npv* has been linked with the industrial LINDO optimization library version 5.3 (Schrage 1995) in order to rely on the procedure of Erengüç et al. (1993). However, this procedure does only lead to optimal solutions when the project deadline δ_n is larger than or equal to the maximal project deadline (i.e., when all activities are scheduled at their normal duration). In this case, a deadline is required when net cash flows are negative, to prevent the total project duration that is equal to infinity. However, problem instances with a lower deadline could not be solved to optimality. To overcome this shortcoming, we have added an extra dummy end node $n + 1$ connected to the last event node n of the project by means of a dummy activity $(n, n + 1)$. Moreover, we have added an extra arc from the start event 1 to the new end dummy event $n + 1$ with a single activity duration $d_{1,n+1}(1) = \delta_n$ with a corresponding $c_{1,n+1}(1) = 0$. A large positive cash flow $C^+_{n+1} = \infty$ is used for the end dummy node to force this event node to finish as soon as possible. The activity $(1, n + 1)$ guarantees that this event node, and consequently the total project, cannot finish earlier than the project deadline δ_n. Both the extra cash flow and the extra arc $(1, n + 1)$ force the project to finish exactly on the project deadline. In doing so, project deadlines smaller than the maximal project deadline are possible—by modifying the duration of activity $(1, n + 1)$—without losing optimality.

4.1 Test settings

The subset of our computational results section is the test set of Demeulemeester et al. (1998) and is used to compare the solutions obtained by our meta-heuristic algorithm with exact solutions for all four problem types. Table 1 displays the parameter settings for the different problem instances of our test set.

This problem set consists of activity-on-the-node networks with different values for the number of activities and the coefficient of network complexity *CNC* and has been generated with the problem generator *ProGen* (Kolisch et al. 1995). Afterwards, these problem instances have been extended with modes for each activity and transformed from activity-on-the-node networks to activity-on-the-arc networks using the algorithm of Kamburowski et al. (1992). Since we study the time/cost trade-off problem under the deadline version, we have extended each problem instance with a project deadline. The project deadline δ_n has been generated as follows. Firstly, we calculate the largest critical path length with every activity at its normal duration. Secondly, we compute the smallest critical path length with every activity at its crash duration. Finally, we set the project deadline δ_n equal to the smallest critical path length exceeded with k_1 times the difference between the largest and the smallest critical path lengths (with k_1 given in Table 1). Consequently, instances with $k_1 = 1$ have a deadline equal to the largest critical path length, while instances with $k_1 = 0$ have a project deadline equal to the smallest critical path length. The details of the instances we have used are displayed in Table 1. The instances as mentioned above can be used to solve the *DTCTP* and results in 4,500 problem instances (using 10 problem instances for each setting). In order to use these instances for solving the *DTCTP-tsc*, the

Table 1 Parameter settings used to generate the test instances

Activity-on-the-Arc Project Scheduling Problems	
DTCTP, DTCTP-tsc, DTCTP-wc, DTCTP-npv	
Number of activities (non-dummy arcs)	10, 20, 30, 40 or 50
Number of modes	Fixed at 2, 4 or 6 or randomly chosen from the interval [1,3], [1,7] or [1,11]
Coefficient of network complexity *CNC*	1.5; 1.8 or 2.1
Constant k_1 for the deadline setting of the project	0, 0.25, 0.50, 0.75 or 1
DTCTP-tsc	
Pattern	[0,0,100], [0,33,66], [0,66,33], [0,100,0], [33,0,66], [33,33,33], [33,66,0], [66,0,33],
[%*day*-pattern, %*d&n*-pattern,%*dnw*-pattern]	[66,33, 0], [100,0,0],
DTCTP-wc	
%act	0.25, 0.50, 0.75
%cost	0.75, 1, 1.25
DTCTP-npv	
Constant k_2 for cash-inflows of events	0, 0.25, 0.5, 0.75, 1

DTCTP-wc and the *DTCTP-npv*, we have generated extra data for each problem instance as shown in Table 1.

In order to solve the *DTCTP-tsc*, each non-dummy activity needs to follow one of its possible patterns. The settings for the pattern, either *day*, *d&n* or *dnw*, of each activity varies as follows: [% of activities following the *day*-pattern, % of activities following the *d&n*-pattern, % of activities following the *dnw*-pattern]. This is exactly the same approach taken by Vanhoucke et al. (2002) and Vanhoucke (2005) to solve the problem to optimality. We rely on the latter procedure to solve this problem type to optimality and to compare it with the meta-heuristic algorithm. Using 10 settings for the time-switch pattern, we have 45,000 problem instances for the *DTCTP-tsc*.

In order to solve the *DTCTP-wc*, the problem instances need to be extended with information about the size of each activity group and its corresponding work continuity cost. The row (of Table 1) labeled "*%act*" denotes the percentage of activities that belongs to the activity group A' expressed as a fraction of the total number of activities in the network, i.e., $|A'| = \%act^*|A|$. The row labeled "*%cost*" is used to determine the work continuity cost w_c, i.e., $w_c = \%cost^* avg\_cost$. The average cost *avg_cost* has been calculated as the average cost slope of all the modes of the activities in A'. In doing so, we can vary *%cost* to influence the relative importance of work continuity versus activity crashing. More precisely, when *%cost* is larger than 1, the idle time cost w_c tends to be more important than the cost of activity crashing, and vice versa. Given our test setting, we have 40,500 problem instances for the *DTCTP-wc*.

The *DTCTP-npv* involves the generation of positive cash flows for the project events. The cash flows $C_j^+ \geq 0$ for each event j have been generated as follows. First, we calculate the minimal cost $C_j^{\min}$ of each event j as the sum of the costs of all incoming arcs of event j with each arc scheduled at its normal duration. Then, we compute the maximal cost $C_j^{\max}$ of each event j as the sum of the costs of all incoming arcs of event j with each arc scheduled at its crash duration. At last, we set the positive cash flow C_j^+ somewhere between the minimal and maximal cost of the event, i.e., $C_j^+ = C_j^{\min} + k_2 * (C_j^{\max} - C_j^{\min})$. Consequently, instances with $k_2 = 0$ will only have events with a zero or negative net cash flow $cf_j(k)$, while instances with $k_2 = 1$ will only have events with a zero or positive net cash flow. Intermediate values of k_2 will result in project network events with both positive and negative net cash flows. Given the settings of k_2 as given in Table 1, we have 22,500 problem instances for the *DTCTP-npv*.

4.2 Experimental results

4.2.1 *Interpretation of the tables*

In the following tables, we display the results of our experimental tests for the different problem types under study. The columns labeled with "*opt*" are used to refer to the results of an exact procedure, while the label "*heur*" is used to refer to the meta-heuristic search procedure results. The column labeled "*Avg CPU*" contains the average CPU time needed to solve the problem instances. The three columns are further subdivided in order to compare the heuristic and exact solution, as follows:

The results for the exact solution procedure are obtained by allowing a maximal time limit of 1 minute. After that, the procedure stops and the solution is reported. In doing so, the obtained solution can be classified in one of the following categories: optimal solution, feasible (but not necessarily optimal) solution or infeasible solution (i.e., no solution found). The columns with label "*% opt*" are used to denote the percentage of problem instances for which an optimal solution has been found. The columns labeled with "*% limit*" display the percentage of problem instances for which a feasible (but not guaranteed to be optimal) solution has been found within the pre-specified time limit of 1 minute. This means that the procedure already has found one or more feasible solutions, but it is truncated after the pre-specified time. The columns with "*% infeas*" show the percentage of problem instances for which no feasible solution has been found within the pre-specified time limit of 1 minute. Each problem instance belongs to one of these three categories, which are used for comparison purposes with the heuristic procedures.

The results found by the heuristic procedure are compared with the results of one of the three categories. The instances for which an exact solution has been found (i.e., "*% opt*") are used to compare them with the heuristic solutions as follows. The column labeled with "*% opt=*" displays the percentage of problem instances for which the heuristic solution has found the optimal solution (only for the problem instances of column "*% opt*"). The column indicated with "*avg opt*" gives the average percentage of deviation from the optimal solution (only for the problem instances of column "*% opt*"). The problem instances with a feasible though not necessarily an optimal solution (i.e., "*% limit*") are analyzed as follows. On the one hand, the column labeled with "*% limit*$\downarrow$" displays the percentage of problem instances for which the heuristic solution is better than the feasible solution found by the exact procedure (only for the problem instances of column "*% limit*"). On the other hand, the column labeled with "*% limit*$\uparrow$" indicates the percentage of problem instances for which the heuristic solution is worse than the feasible solution found by the exact procedure (only for the problem instances of column "*% limit*"). The remaining fraction is then the percentage of problem instances with a solution equal to the feasible solution found. Furthermore, the columns labeled with "*avg limit*$\downarrow$" display the average percentage of deviation (improvement) of the heuristic solution (only for the problem instances of column "*% limit*"), while the columns with label "*avg limit*$\uparrow$" refer to the average percentage of deviation (deterioration) of the heuristic solution (only for the problem instances of column "*% limit*").

The two symbols, sol_H and sol_O, are used to denote the solution found by, respectively, the meta-heuristic procedure and the exact procedure for the problem type under study. Deviations between the solutions found by the exact algorithm and the meta-heuristic approach are calculated as follows:

1. The *average deviation from the optimal solution* is calculated as $avg\,opt = \frac{|sol_H - sol_O|}{sol_O} * 100$ and is only calculated for the instances for which an optimal solution has been found (i.e., column "*% opt*").
2. The *average improvement* from the truncated solution from the exact algorithm is calculated as $avg\,limit\!\downarrow = \frac{|sol_O - sol_H|}{sol_O} * 100$ and is only calculated for the instances of column "*% limit*". Note that the symbol $\downarrow$ is used to refer to improved solutions, and not a lower objective function value. Consequently, an improved solution for the *DTCTP*, *DTCTP-wc,* and *DTCTP-tsc* is a solution with a *lower* cost, while an improved solution for the *DTCTP-npv* is a solution with a *higher* net present value.
3. *Average deterioration* from the truncated solution from the exact algorithm is calculated as $avg\,limit\!\uparrow = \frac{|sol_H - sol_O|}{sol_O} * 100$ and is only calculated for the instances of column "*% limit*".

The columns labeled "*Created nodes*" have a different interpretation depending on the problem type. The number of created nodes is used to refer to the number of nodes in the branch-and-bound tree for the optimal solution procedures of the *DTCTP*, *DTCTP-tsc,* and *DTCTP-wc*, while it equals the number of iterations between main- and sub-problem for the benders decomposition approach of the *DTCTP-npv*. The number of created nodes is equal to the number of visited solutions in the meta-heuristic search for all problem types. For this reason, the values for these columns serve only to describe the hardness of the problem instances, but cannot be compared with each other.

The columns labeled "DP" are used to display the number of improvement scenarios evaluated per dynamic programming step. Indeed, the dynamic programming procedure is stopped after a pre-specified number of backtracking steps (parameter par_4) and consequently, only a subset (i.e., DP) of the possible improvement scenarios is evaluated.

We have fine-tuned the settings for the different parameters, and have finally selected the following values: $par_1 = 25$, $par_2 = 7$, $par_3 = 0$, and $par_4 = 10$.

4.2.2 *General experimental results for all problem types*

The overall results are very encouraging, as displayed in the different tables of the four problem types. The *DTCTP* and the *DTCTP-wc* can be solved very efficiently to optimality and, therefore, there is no need to rely on heuristic procedures. These meta-heuristics, however, are able to find an optimal solution in almost all the cases (92.13% for *DTCTP* and 83.38% for the *DTCTP-wc*), with a small deviation from the optimal solutions when this is not the case. However, the CPU-time is larger than the one for the optimal procedures and, therefore, one does not need to rely on meta-heuristic procedures for these problem types. However, the meta-heuristic procedure has its merits when solving the *DTCTP-tsc* and the *DTCTP-npv*, as described in Sects. 4.2.4 and 4.2.5. In the sequel of this section, we describe some computational results in more detail. In the next section, we elaborate on problem-type specific characteristics of both the optimal and heuristic procedures.

For the rows labeled '*number of activities*', '*number of modes*', '*CNC*', and '*pattern*', the results mainly correspond to the results found by Demeulemeester et al. (1998) and confirmed by Vanhoucke et al. (2002), and Vanhoucke (2005). The negative effect of both the *'number of activities'* and the '*number of modes*' on the problem complexity is straightforward and can be seen in all tables, as follows:

- *Computational effort*: increasing "*Avg CPU*" for all optimal and heuristic procedures.
- *Percentage of instances solved to optimality*: decreasing "*% opt*" (for the optimal procedures) and "*% opt* =" (for the heuristic procedures). This positive effect is not so clear for the heuristic procedure for the *DTCTP-npv*, since the column "*% opt* =" reveals that the number of instances solved to optimality with the meta-heuristic procedure shows some counterintuitive results for 40 activities and 6 or $[1,11]$ modes.
- *Average deviation from the optimal solution*: Increasing values for the "*avg opt*" columns for all but one meta-heuristic procedure. Indeed, only the *DTCTP-npv* shows some counterintuitive results. However, only a small subpart of the problem set can be solved to optimality, when the problem size (both in terms of activities and number of modes) increases. The percentage of problems solved to optimality even drops to 19.92% for instances with 6 modes. Therefore, the percentage deviation could only be measured for a small part of the subset and might lead to biased results.
- *Created nodes*: Increasing number of nodes for all optimal and heuristic procedures. As mentioned earlier, these values are only used to describe the hardness of the problem instances, but cannot be compared with each other.

In line with literature, we observe however that problem instances with a fixed number of modes (2 or 4) are more difficult to solve than the instances where the number of modes is randomly selected (between $[1,3]$ or $[1,7]$). Although the average number of modes per activity remains the same (e.g., $[1,3]$ corresponds to—on the average—2 modes), variation in the number of modes between different activities has a positive effect on the problem complexity.

The effect of the *CNC* is not so clear for both the optimal and heuristic procedures. The average CPU time seems to increase with larger values for the CNC, except for the *DTCTP-npv*. The number of created nodes shows similar results for most optimal procedures (except for the *DTCTP-tsc*), but shows an opposite result for the meta-heuristic procedures (or no relation for the *DTCTP-wc*). However, it has already been shown extensively in the literature that the *CNC* is not a good measure to predict the difficulty of project scheduling problems (see, e.g., Elmaghraby and Herroelen 1980; De Reyck and Herroelen 1996, and Herroelen and De Reyck 1999) and, therefore, the results needs to be interpreted with care.

The results in the row labeled '*deadline*' indicate the effect of the project deadline on the problem complexity and is in line with the results found in the literature. All tables reveal a negative relation between the project deadline and problem complexity, i.e., the larger the project deadline (larger k_1 values), the easier the problem, which can be seen in "*Avg CPU*", "*Created nodes*", and "*% opt* =". Only project instances with a deadline equal to the smallest critical path length (i.e., with $k=0$) are often easy to solve. However, these results from the literature could not be confirmed for the *DTCTP-npv*.

4.2.3 *Experimental results for the DTCTP and the DTCTP-wc*

Tables 2 and 3 summarize the computational results for the *DTCTP* and the *DTCTP-wc*, respectively. Since there is no major difference between the optimal procedures for both problem types (apart from two extra arcs), we discuss the results together. Both tables reveal that the exact algorithm is very efficient for the problem types and outperforms our meta-heuristic, both in terms of solution quality and computational effort. Indeed, although the meta-heuristic procedure is able to solve 92.13% (83.38%) of the problem instances to optimality for the *DTCTP* (*DTCTP-wc*), the computational effort is much larger than for the exact algorithm. Therefore, we conclude that it is not beneficial to rely on this meta-heuristic procedure to solve instances of this size.

Remark that all problem instances could be solved to optimality with the exact algorithm, and, therefore, no values have been reported for the columns "*% limit*" and "*% infeas*". Consequently, for obvious reasons, the columns "*% limit*↓" and "*% limit*↑", and their corresponding columns "*avg limit*↓" and "*avg limit*↑" are also empty.

It is tempting to conclude that the meta-heuristic might outperform the exact algorithm, when the problem instances increase in size and pass a certain threshold. Therefore, we have generated a second dataset. More precisely, we have generated project networks with 100, 150, and 200 non-dummy activities and with the number of modes generated randomly from the interval $[1,10]$ and $[1,50]$. All other settings are similar than in Table 1. Using 10 instances for each setting, we obtain 2,700 instances for the *DTCTP* and 24,300 instances for the *DTCTP-wc*. We compared the results of the meta-heuristic procedure with a truncated version of the branch-and-bound procedure (truncated after 60 seconds). The results revealed that the truncated exact

Table 2 Computational results for the *DTCTP*

DTCTP	*Opt*			*Heur*						*Avg CPU*		*Created nodes*		*DP*
	% opt	*% limit*	*% infeas*	*% opt =*	*avg opt*	*% limit↓*	*avg limit↓*	*% limit↑*	*avg limit↑*	*opt*	*heur*	*opt*	*heur*	*heur*
overall	100%	0%	0%	92.13%	0.068%	–	–	–	–	0.081	0.571	352	5,508	17
act														
10	100%	0%	0%	99.00%	0.037%	–	–	–	–	0.000	0.008	15	1,123	8
20	100%	0%	0%	96.67%	0.050%	–	–	–	–	0.002	0.096	59	3,008	11
30	100%	0%	0%	94.78%	0.044%	–	–	–	–	0.011	0.337	144	5,387	14
40	100%	0%	0%	86.78%	0.098%	–	–	–	–	0.058	0.811	349	7,892	17
50	100%	0%	0%	83.44%	0.114%	–	–	–	–	0.333	1.605	1,195	10,130	20
mod														
2	100%	0%	0%	95.47%	0.040%	–	–	–	–	0.002	0.102	30	1,543	10
4	100%	0%	0%	90.13%	0.102%	–	–	–	–	0.057	0.554	318	5,539	16
6	100%	0%	0%	85.47%	0.133%	–	–	–	–	0.273	1.138	1,034	9,335	20
[1, 3]	100%	0%	0%	98.80%	0.006%	–	–	–	–	0.001	0.084	18	1,501	8
[1, 7]	100%	0%	0%	93.87%	0.047%	–	–	–	–	0.022	0.478	153	5,374	14
[1, 11]	100%	0%	0%	89.07%	0.082%	–	–	–	–	0.129	1.072	561	9,759	18
CNC														
1.5	100%	0%	0%	92.33%	0.060%	–	–	–	–	0.035	0.425	258	5,640	17
1.8	100%	0%	0%	92.73%	0.062%	–	–	–	–	0.077	0.568	322	5,495	16
2.1	100%	0%	0%	91.33%	0.083%	–	–	–	–	0.131	0.721	478	5,389	17
k_1														
0	100%	0%	0%	94.22%	2.900%	–	–	–	–	0.163	0.676	505	2,377	47
0.25	100%	0%	0%	81.44%	17.800%	–	–	–	–	0.221	0.977	1,011	5,507	29
0.5	100%	0%	0%	88.11%	10.200%	–	–	–	–	0.018	0.692	198	6,497	17
0.75	100%	0%	0%	96.89%	3.300%	–	–	–	–	0.002	0.384	47	6,642	9
1	100%	0%	0%	100%	0.000%	–	–	–	–	0.000	0.128	1	6,519	2

algorithm still outperforms the meta-heuristic procedures. The reason is that the specific approach used for the exact branch-and-bound algorithm of Demeulemeester et al. (1998) results very quickly in truncated (heuristic) solutions that are very close to the optimal solution. Moreover, thanks to the use of an efficient lower bound calculation of Ford and Fulkerson (1962), many nodes can be evaluated in the branch-and-bound tree within a limited amount of CPU-time, and hence, the meta-heuristic procedure has no computational advantage on that aspect. Therefore, we conclude that, due to the efficient character of the exact algorithm of Demeulemeester et al. (1998), the meta-heuristic solutions can not compete with a truncated solutions found by the exact algorithm.

4.2.4 *Experimental results for the DTCTP-tsc*

Table 4 displays the results for the *DTCTP-tsc* and shows that the meta-heuristic algorithm is a good alternative to the exact algorithm of Vanhoucke (2005). 89.76% of the problem instances could be solved to optimality with the exact procedure, from which 96.36% were found to be optimal by the meta-heuristic approach within a much lower computational effort. The remaining 10.24% could not be solved to optimality with the exact procedure (in 2.16% of the cases, the exact algorithm was even not able to find a feasible solution within the 60 seconds time limit). In 56.74% of the cases, the meta-heuristic approach was able to improve the truncated solutions by—on the average—1.303%. The overall CPU time is—on the average—only a fraction (about 12%) of the CPU time for the exact algorithm. Therefore, we claim that the meta-heuristic procedure outperforms the exact algorithm, since it is able to find near-optimal solutions (or even improved solutions) in a reasonable time limit. The detailed results for the different parameters are in line with literature and are displayed, without any further discussion, in Table 4.

4.2.5 *Experimental results for the DTCTP-npv*

Table 5 displays the results for the *DTCTP-npv* and reveals that a meta-heuristic algorithm is necessary to find near-optimal solutions. The algorithm of Erengüç et al. (1993) relies on the Benders Decomposition approach and

Table 3 Computational results for the *DTCTP-wc*

DTCTP-wc	*Opt*			*Heur*						*Avg CPU*		*Created nodes*		*DP*
	% opt	*% limit*	*% infeas*	*% opt =*	*avg opt*	*% limit↓*	*avg limit↓*	*% limit↑*	*avg limit↑*	*opt*	*heur*	*opt*	*heur*	*heur*
overall	100%	0%	0%	83.38%	0.165%	–	–	–	–	0.105	1.220	343	8,514	24
act														
10	100%	0%	0%	97.58%	0.065%	–	–	–	–	0.020	0.028	14	2,345	13
20	100%	0%	0%	90.23%	0.135%	–	–	–	–	0.022	0.289	56	5,457	19
30	100%	0%	0%	83.11%	0.189%	–	–	–	–	0.034	0.845	152	8,543	22
40	100%	0%	0%	75.95%	0.188%	–	–	–	–	0.090	1.801	377	11,738	26
50	100%	0%	0%	70.01%	0.247%	–	–	–	–	0.362	3.135	1,117	14,487	28
mod														
2	100%	0%	0%	92.80%	0.065%	–	–	–	–	0.022	0.438	28	3,652	20
4	100%	0%	0%	78.77%	0.222%	–	–	–	–	0.090	1.311	337	8,850	26
6	100%	0%	0%	71.72%	0.296%	–	–	–	–	0.312	2.229	1,016	13,474	28
[1, 3]	100%	0%	0%	96.77%	0.026%	–	–	–	–	0.021	0.324	17	3,412	16
[1, 7]	100%	0%	0%	83.79%	0.122%	–	–	–	–	0.047	1.091	172	8,392	22
[1, 11]	100%	0%	0%	76.41%	0.259%	–	–	–	–	0.141	1.924	491	13,304	25
CNC														
1.5	100%	0%	0%	83.48%	0.170%	–	–	–	–	0.059	0.850	260	8,411	24
1.8	100%	0%	0%	83.69%	0.153%	–	–	–	–	0.093	1.279	296	8,662	25
2.1	100%	0%	0%	82.96%	0.172%	–	–	–	–	0.165	1.530	474	8,469	25
k_1														
0	100%	0%	0%	93.16%	0.040%	–	–	–	–	0.183	0.757	485	2,718	46
0.25	100%	0%	0%	73.72%	0.314%	–	–	–	–	0.251	1.453	943	7,364	33
0.5	100%	0%	0%	80.74%	0.181%	–	–	–	–	0.049	1.399	224	9,674	24
0.75	100%	0%	0%	86.80%	0.096%	–	–	–	–	0.023	1.223	48	10,881	19
1	100%	0%	0%	82.47%	0.194%	–	–	–	–	0.021	1.266	17	11,932	19
%act														
0.25	100%	0%	0%	86.33%	0.143%	–	–	–	–	0.101	1.295	320	9,201	24
0.5	100%	0%	0%	83.59%	0.164%	–	–	–	–	0.106	1.226	348	8,543	25
0.75	100%	0%	0%	80.21%	0.189%	–	–	–	–	0.109	1.138	362	7,797	25
%cost														
0.75	100%	0%	0%	86.38%	0.140%	–	–	–	–	0.105	1.103	341	8,189	23
1	100%	0%	0%	83.43%	0.160%	–	–	–	–	0.106	1.220	342	8,528	24
1.25	100%	0%	0%	80.33%	0.195%	–	–	–	–	0.106	1.335	347	8,824	26

is only able to solve small problem instances. 49.70% of the problem instances could be solved to optimality with the exact procedure, from which 90.04% were found to be optimal by the meta-heuristic approach within a much lower computational effort (32 seconds versus 3 seconds). Project instances with up to 50 activities can only be solved to optimality in 33.93% of the cases. 50.30% of all instances could not be solved to optimality with the exact procedure. In 86.59% of the cases, the meta-heuristic approach was able to improve the truncated solutions by—on the average—137.221%. This large improvement is due to the fact that the BD approach often needs more than 60 seconds to solve the time-consuming LP model in the first iteration, and, therefore, no good solutions can be found.

The overall CPU time is—on the average—only a small fraction (about 10%) of the CPU time for the exact algorithm. Therefore, we claim that the meta-heuristic procedure outperforms the exact algorithm, since it is able to find near-optimal solutions and, in many cases, strongly improved solutions in a reasonable time limit. The detailed results for the different parameters are in line with literature and are displayed, without any further discussion, in Table 5.

The solution found for the *DTCTP-npv* depends on the deadline (k_1) and the cost (k_2) of the project instance, and

Table 4 Computational results for the *DTCTP-tsc*

DTCTP-tsc	*Opt*			*Heur*						*Avg CPU*		*Created nodes*		*DP*
	% opt	*% limit*	*% infeas*	*% opt =*	*avg opt*	*% limit↓*	*avg limit↓*	*% limit↑*	*avg limit↑*	*opt*	*heur*	*opt*	*heur*	*heur*
overall	89.76%	8.08%	2.16%	96.36%	0.039%	56.74%	1.303%	9.74%	0.808%	7.906	0.926	290	5,562	16
act														
10	100.00%	0.00%	0.00%	99.49%	0.023%	–	–	–	–	0.004	0.019	4	1,171	7
20	99.78%	0.22%	0.00%	97.35%	0.045%	40.00%	1.440%	10.00%	1.349%	0.537	0.169	64	3,092	10
30	95.11%	4.73%	0.16%	95.81%	0.038%	47.18%	1.298%	7.51%	0.825%	5.064	0.562	250	5,460	13
40	82.34%	15.31%	2.34%	93.79%	0.049%	58.13%	1.296%	9.14%	0.820%	13.549	1.33	471	7,927	16
50	71.56%	20.13%	8.31%	94.29%	0.043%	58.14%	1.308%	10.71%	0.791%	20.379	2.548	660	10,160	19
mod														
2	99.28%	0.71%	0.01%	96.82%	0.033%	35.85%	1.508%	5.66%	1.296%	1.134	0.167	174	1,560	9
4	87.91%	10.31%	1.79%	94.75%	0.065%	57.83%	1.246%	8.80%	0.780%	9.676	0.895	313	5,589	15
6	78.19%	14.81%	7.00%	94.00%	0.071%	56.53%	1.393%	11.16%	0.934%	15.498	1.866	315	9,434	19
[1, 3]	99.25%	0.64%	0.11%	99.30%	0.007%	31.25%	1.072%	0.00%	–	1.012	0.138	186	1,518	7
[1, 7]	90.93%	8.21%	0.85%	97.16%	0.022%	57.47%	1.297%	5.84%	0.708%	7.422	0.776	344	5,445	13
[1, 11]	82.99%	13.80%	3.21%	95.34%	0.047%	57.97%	1.254%	11.88%	0.713%	12.697	1.711	407	9,826	17
CNC														
1.5	90.82%	7.63%	1.55%	96.38%	0.037%	57.90%	1.307%	10.13%	0.820%	7.059	0.739	297	5,682	15
1.8	89.88%	8.25%	1.87%	96.51%	0.035%	56.35%	1.275%	9.46%	0.701%	7.884	0.923	317	5,546	15
2.1	88.57%	8.36%	3.07%	96.18%	0.045%	56.06%	1.326%	9.65%	0.899%	8.776	1.115	255	5,458	16
k_1														
0	81.18%	11.61%	7.21%	97.86%	0.013%	50.53%	0.947%	3.25%	0.498%	13.909	1.115	210	2,584	45
0.25	78.77%	17.77%	3.47%	91.75%	0.100%	62.04%	1.409%	12.63%	0.826%	15.569	1.583	426	5,673	27
0.5	90.39%	9.48%	0.13%	93.28%	0.072%	54.87%	1.453%	13.01%	0.879%	8.139	1.095	492	6,466	15
0.75	98.46%	1.54%	0.00%	97.93%	0.023%	53.96%	1.466%	5.04%	0.643%	1.910	0.620	320	6,569	8
1	100.00%	0.00%	0.00%	100.00%	0.000%	–	–	–	–	0.004	0.216	1	6,519	2
pattern														
[0, 0, 100]	95.58%	3.29%	1.13%	95.70%	0.046%	40.54%	20.95%	%0.827%	0.827%	4.074	0.903	52	5,636	15
[0, 33, 66]	96.84%	2.51%	0.64%	94.40%	0.062%	35.40%	0.756%	20.35%	0.894%	3.225	0.935	27	5,515	16
[0, 66, 33]	86.80%	10.71%	2.49%	96.70%	0.036%	64.52%	1.462%	7.68%	0.842%	9.837	0.94	325	5,500	16
[0, 100, 0]	81.69%	14.47%	3.84%	96.98%	0.035%	62.83%	1.588%	5.38%	0.709%	13.387	0.953	752	5,525	17
[33, 0, 66]	98.42%	1.22%	0.36%	94.97%	0.053%	32.73%	0.677%	23.64%	0.848%	1.898	0.857	14	5,605	14
[33, 33, 33]	91.60%	6.73%	1.67%	97.67%	0.020%	54.79%	1.114%	12.21%	0.888%	6.597	0.901	163	5,617	15
[33, 66, 0]	85.00%	12.18%	2.82%	97.41%	0.031%	57.12%	1.383%	8.21%	0.867%	11.163	0.929	545	5,576	15
[66, 0, 33]	90.29%	7.71%	2.00%	96.80%	0.033%	51.30%	1.086%	8.36%	0.811%	7.532	0.91	191	5,561	15
[66, 33, 0]	84.73%	12.02%	3.24%	97.25%	0.032%	58.96%	1.189%	9.80%	0.780%	11.195	0.955	491	5,562	16
[100, 0, 0]	86.62%	9.96%	3.42%	96.38%	0.039%	55.58%	1.178%	11.38%	0.706%	10.157	0.973	337	5,523	17

has been displayed in Table 6. This table gives the heuristic solution found by the meta-heuristic procedure for different values for k_1 and k_2. The row with $k_2 = 0$ displays only negative values for the *npv*, apart from the $k_1 = 1$ column. When $k_2 = 0$, $C_j^+ = C_j^{\min}$ and, hence, the best possible solution that can be obtained is with normal durations for each activity (then the cash flow of each event equals exactly the cost of all incoming arcs). This is only possible with a project deadline equal to its maximal value (i.e., $k_1 = 1$). Decreasing that project deadline results in activity costs exceeding the cash inflow of each event, and eventually, a negative net present value. Larger values for k_2 result in larger values for the cash inflow C_j^+ of each event j and can possibly lead to positive net present values, depending on the project deadline.

5 Conclusions and suggestions for future research

In this paper we have presented a detailed description of the well-known discrete time/cost trade-off problem

Table 5 Computational results for the *DTCTP-npv*

DTCTP-npv	*Opt*			*Heur*						*Avg CPU*		*Created nodes*		*DP*
	% opt	*% limit*	*% infeas*	*% opt =*	*avg opt*	*% limit↓*	*avg limit↓*	*% limit↑*	*avg limit↑*	*opt*	*heur*	*opt*	*heur*	*heur*
overall	49.70%	50.30%	–	90.04%	0.248%	86.59%	137.221%	6.82%	6.676%	32.948	3.160	17	6,437	36
act														
10	79.13%	20.87%	–	95.62%	0.117%	78.27%	196.900%	6.50%	15.095%	14.798	0.046	12	1,113	12
20	53.96%	46.04%	–	89.74%	0.109%	86.25%	239.477%	7.67%	13.626%	30.942	0.465	18	3,205	21
30	42.40%	57.60%	–	87.84%	0.413%	87.42%	117.355%	7.45%	4.492%	37.140	1.750	19	6,005	29
40	39.07%	60.93%	–	88.00%	0.466%	87.89%	116.382%	6.56%	3.836%	39.232	4.402	18	9,257	37
50	33.93%	66.07%	–	82.58%	0.320%	87.52%	86.737%	6.02%	2.844%	42.629	9.136	19	12,604	46
mod														
2	87.23%	12.77%	–	87.83%	0.188%	71.40%	38.184%	5.64%	71.688%	12.263	0.445	18	1,509	17
4	33.17%	66.83%	–	84.08%	0.326%	83.56%	112.738%	9.06%	5.353%	43.234	2.881	21	6,138	35
6	19.92%	80.08%	–	90.90%	0.156%	89.91%	170.485%	5.63%	2.836%	49.838	6.483	16	11,625	43
[1, 3]	91.95%	8.05%	–	94.17%	0.396%	76.82%	88.321%	2.65%	53.650%	6.922	0.408	11	1,464	14
[1, 7]	44.59%	55.41%	–	88.88%	0.133%	84.31%	133.446%	9.48%	3.142%	36.280	2.655	19	6,017	31
[1, 11]	21.33%	78.67%	–	92.13%	0.066%	90.85%	142.177%	4.88%	3.304%	49.152	6.087	17	11,868	38
CNC														
1.5	48.48%	51.52%	–	89.71%	0.212%	85.66%	102.104%	6.55%	3.740%	33.341	2.310	16	6,607	36
1.8	49.45%	50.55%	–	88.86%	0.292%	86.55%	149.283%	7.36%	9.300%	33.102	3.205	17	6,430	36
2.1	51.16%	48.84%	–	91.48%	0.241%	87.61%	161.111%	6.55%	6.720%	32.402	3.965	18	6,274	37
k_1														
0	46.93%	53.07%	–	94.03%	0.062%	94.03%	86.818%	0.04%	57.256%	34.459	1.964	11	2,421	73
0.25	30.98%	69.02%	–	84.72%	0.140%	94.59%	218.424%	1.38%	32.298%	43.677	4.339	19	6,056	61
0.5	40.98%	59.02%	–	75.98%	0.169%	92.55%	177.359%	3.16%	26.095%	39.343	3.805	25	7,363	41
0.75	53.98%	46.02%	–	78.76%	0.851%	70.88%	43.312%	13.33%	3.637%	31.555	2.934	23	7,925	25
1	75.62%	24.38%	–	85.81%	0.021%	56.43%	3.793%	33.55%	1.391%	15.707	2.757	9	8,419	15
k_2														
0	53.47%	46.53%	–	97.96%	0.911%	88.20%	56.366%	0.72%	53.419%	30.723	3.237	17	5,527	34
0.25	54.78%	45.22%	–	96.59%	0.161%	91.60%	170.155%	1.23%	92.597%	30.274	2.683	18	5,697	31
0.5	51.11%	48.89%	–	90.78%	0.046%	86.36%	237.051%	5.36%	4.531%	32.228	2.740	17	6,121	34
0.75	45.58%	54.42%	–	84.89%	0.028%	83.79%	186.455%	11.31%	3.047%	35.262	3.184	17	6,877	38
1	43.56%	56.44%	–	76.58%	0.011%	84.13%	42.343%	13.27%	1.955%	36.254	3.955	16	7,962	42

Table 6 The impact of k_1 and k_2 on the net present value for the *DTCTP-npv*

k_2	k_1				
	0	0.25	0.5	0.75	1
0	−6878.82	−1122.62	−198.81	−39.05	0.00
0.25	−4700.87	−177.58	497.67	603.64	629.29
0.5	−2122.80	1069.15	1458.05	1510.32	1523.34
0.75	322.43	2430.74	2612.63	2627.15	2631.09
1	3254.43	4440.14	4481.88	4471.63	4462.44

(*DTCTP*) as follows: the basic *DTCTP*, the *DTCTP* with time-switch constraints (*tsc*), the *DTCTP* with work continuity constraints (*wc*), and the *DTCTP* with net present value maximization (*npv*). We gave an extensive literature overview of existing procedures for these four problem types and computed exact solutions for the four versions of the discrete time/cost trade-off problem. Moreover, we developed a new meta-heuristic approach in order to provide near-optimal heuristic solutions for the different problems under study. Finally, we presented computational results for the problems under study by comparing the results for both exact and heuristic procedures. The results can be summarized per problem type as follows.

The literature for the *DTCTP* is rich and many procedures are able to produce exact results in an efficient way. In this paper we relied on the procedure developed by Demeulemeester et al. (1998), since this is—to the best of our knowledge—the best performing algorithm in the literature. Although the meta-heuristic procedure is able to produce consistently good results, it cannot outperform the exact procedure and, therefore, it is not a good alternative to the exact approaches for the *DTCTP*.

The literature for the *DTCTP-wc* is virtually void. Reda (1990) has discussed a slightly different version of this problem type, but did not report any computational results. Although the importance of work continuity constraints has been recognized in the literature for several decades, no efficient algorithm has been presented to solve the *DTCTP-wc* to optimality. In this paper we have extended the basic *DTCTP* to the *DTCTP-wc* by adding two extra arcs per work continuity constraint. In doing so, we recognized the trade-off between idle minimization of resources (work continuity) and direct activity cost (time/cost behaviour). As for the *DTCTP*, the meta-heuristic procedure is not able to outperform the exact algorithms, due to the efficient character of the latter procedures.

The literature on exact algorithms for the *DTCTP-tsc* is restricted to Vanhoucke et al. (2002) and Vanhoucke (2005). Both procedures consist of a double branch-and-bound to solve the problem instances to optimality. The meta-heuristic procedure can generate near-optimal solutions within a reasonable time, and is an excellent alternative to the exact algorithms. More precisely, the meta-heuristic algorithm can solve almost all instances (96.36%) to optimality within a very small computational time. The instances that cannot be solved to optimality with exact procedures can be improved by the meta-heuristic procedure in 56.74% of the cases. The time is only a fraction of the time needed by the exact algorithms.

The *DTCTP-npv* literature is restricted to Erengüç et al. (1993). Due to the inefficient character of this solution approach, meta-heuristics are able to outperform these solutions dramatically. More precisely, only 49.70% of the problem instances could be solved to optimality with the exact procedure, from which 90.04% of the solutions were found by the meta-heuristic procedure. 86.59% of all other instances that could not be solved to optimality by the exact procedure could be improved by our meta-heuristic search procedure (with an average improvement of 137.221%).

Our future search efforts will focus on two extensions. First, we will focus on the development of new and better meta-heuristic procedures for the *DTCTP-tsc* and the *DTCTP-npv*. Our results revealed that the exact algorithm for the *DTCTP* and the *DTCTP-wc* performs excellent, and, therefore, we believe that truncated versions of these procedures will produce consistently good results for larger problem instances. However, our results also revealed that the meta-heuristic procedures can outperform the existing exact algorithms for the *DTCTP-tsc* and the *DTCTP-npv*, and, therefore, further research should focus on improving these meta-heuristic search procedures.

A second extension could lie in the adjustment of the *DTCTP* to other realistic settings. We believe that the extension from the well-known *DTCTP* to work continuity, time/switch constraints, and net present value was only the first step in this process. Depending on the needs of companies and project schedulers, further extensions might seem interesting and worth investigating. A straightforward extension could be that the *DTCTP-tsc* is changed, so that each activity can follow an activity pattern (day, day–night or day–night–weekend) but without predefining the activity pattern in advance. In such, each pattern should have a different cost, and the objective is then to assign a pattern to each activity, such that the total cost (i.e., direct activity cost plus the cost of the pattern assigned) is minimized. This straightforward extension of the basic version of the *DTCTP-tsc* would meet the demand of project managers and would require further research, both to exact and (meta-) heuristic procedures.

References

Akkan, C., Drexl, A., & Kimms, A. (2000a). Network decomposition for the discrete time/cost trade-off problem, part 1: models and bounding methods. In Proceedings of the seventh international workshop on project management and scheduling (pp. 29–31), Osnabrück, Germany, 17–19 April 2000.

Akkan, C., Drexl, A., & Kimms, A. (2000b). Network decomposition for the discrete time/cost trade-off problem, part 2: network decomposition and computational results. In Proceedings of the seventh international workshop on project management and scheduling (pp. 32–34), Osnabrück, Germany, 17–19 April 2000.

Bianco, L., & Speranza, M. G. (1990). Resource management in project scheduling. In Proceedings of the second international workshop on project management and scheduling, Compiègne, France, 20–22 June 1990.

Billstein, N., & Radermacher, F. J. (1977). Time-cost optimization. *Methods of Operations Research*, *27*, 274–294.

Chen, Y. L., Rinks, D., & Tang, K. (1997). Critical path in an activity network with time constraints. *European Journal of Operational Research*, *100*, 122–133.

Crowston, W. (1970). Network reduction and solution. *Operations Research Quarterly*, *21*, 435–450.

Crowston, W., & Thompson, G. L. (1967). Decision CPM: a method for simultaneous planning, scheduling and control of projects. *Operations Research*, *15*, 407–426.

De, P., Dunne, E. J., Ghosh, J. B., & Wells, C. E. (1995). The discrete time/cost trade-off problem revisited. *European Journal of Operational Research*, *81*, 225–238.

De, P., Dunne, E. J., Ghosh, J. B., & Wells, C. E. (1997). Complexity of the discrete time/cost trade-off problem for project networks. *Operations Research*, *45*, 302–306.

De Boer, R. (1998). *Resource-constrained multi-project management—a hierarchical decision support system*. PhD dissertation, Institute for Business Engineering and Technology Application, The Netherlands.

De Reyck, B., & Herroelen, W. (1996). On the use of the complexity index as a measure of complexity in activity networks. *European Journal of Operational Research*, *91*, 347–366.

Deineko, V. G., & Woeginger, G. J. (2001). Hardness of approximation of the discrete time/cost trade-off problem. *Operations Research Letters*, *29*, 207–210.

Demeulemeester, E., Elmaghraby, S. E., & Herroelen, W. (1996). Optimal procedures for the discrete time/cost trade-off problem in project networks. *European Journal of Operational Research*, *88*, 50–68.

Demeulemeester, E., De Reyck, B., Foubert, B., Herroelen, W., & Vanhoucke, M. (1998). New computational results for the discrete time/cost trade-off problem in project networks. *Journal of the Operational Research Society*, *49*, 1153–1163.

El-Rayes, K., & Moselhi, O. (1998). Resource-driven scheduling of repetitive activities. *Construction Management and Economics*, *16*, 433–446.

Elmaghraby, S.E., & Herroelen, W. (1980). On the measurement of complexity in activity networks. *European Journal of Operational Research*, *5*, 223–234.

Elmaghraby, S. E., & Kamburowski, J. (1992). The analysis of activity networks under generalized precedence relations. *Management Science*, *38*, 1245–1263.

Elmaghraby, S. E., & Salem, A. (1982). Optimal project compression under quadratic cost functions. *Applications of Management Science*, *2*, 1–39.

Elmaghraby, S. E., & Salem, A. (1984). Optimal linear approximation in project compression. *IIE Transactions*, *16*(4), 339–347.

Erengüç, S. S., Tufekci, S., & Zappe, C. J. (1993). Solving time/cost trade-off problems with discounted cash flows using generalized Benders decomposition. *Naval Research Logistics*, *40*, 25–50.

Falk, J. E., & Horowitz, J. L. (1972). Critical path problems with concave cost-time curves. *Management Science*, *19*, 446–455.

Ford, L. R., & Fulkerson, D. R. (1962). *Flows in networks*. New Jersey: Princeton University Press.

Fulkerson, D. R. (1961). A network flow computation for project cost curves. *Management Science*, *7*, 167–178.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, *5*, 533–549.

Gong, D. (1997). Optimization of float use in risk analysis-based network scheduling. *International Journal of Project Management*, *15*(3), 187–192.

Goto, E., Joko, T., Fujisawa, K., Katoh, N., & Furusaka, S. (2000). *Maximizing net present value for generalized resource constrained project scheduling problem*. Working paper Nomura Research Institute, Japan.

Goyal, S. K. (1975). A note on the paper: a simple CPM time/cost trade-off algorithm. *Management Science*, *21*, 718–722.

Harris, R. B., & Ioannou, P. G. (1998). Scheduling projects with repeating activities. *Journal of Construction Engineering and Management*, *124*(4), 269–278.

Herroelen, W., & De Reyck, B. (1999). Phase transitions in project scheduling. *Journal of Operational Research Society*, *50*, 148–156.

Herroelen, W., Demeulemeester, E., & Van Dommelen, P. (1997). Project network models with discounted cash flows: A guided tour through recent developments. *European Journal of Operational Research*, *100*, 97–121.

Herroelen, W., Demeulemeester, E., & De Reyck, B. (1999). A classification scheme for project scheduling problems. In J. Weglarz (Ed.), *Handbook on recent advances in project scheduling* (Chap. 1, pp. 1–26). Amsterdam: Kluwer Academic.

Hindelang, T. J., & Muth, J. F. (1979). A dynamic programming algorithm for decision CPM networks. *Operations Research*, *27*, 225–241.

Kamburowski, J., Michael, D. J., & Stallmann, M. F. M. (1992). Optimal construction of project activity networks. In *Proceedings of the 1992 annual meeting of the Decision Sciences Institute* (pp. 1424–1426), San Francisco, USA.

Kapur, K. C. (1973). An algorithm for the project cost/duration analysis problem with quadratic and convex cost functions. *IIE Transactions*, *5*, 314–322.

Kelley, J. E. (1961). Critical path planning and scheduling: Mathematical basis. *Operations Research*, *9*, 296–320.

Kelley, J. E., & Walker, M. R. (1959). *Critical path planning and scheduling: an introduction*. Ambler: Mauchly Associates.

Kolisch, R., Sprecher, A., & Drexl, A. (1995). Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science*, *41*, 1693–1703.

Lamberson, L. R., & Hocking, R. R. (1970). Optimum time compression in project scheduling. *Management Science*, *16*, B-597–B-606.

Moder, J. J., Phillips, C. R., & Davis, E. W. (1983). *Project management with CPM, PERT and precedence diagramming* (3rd ed.). New York: Van Nostrand Reinhold Company.

Patterson, J. H., & Harvey, R. T. (1979). An implicit enumeration algorithm for the time/cost trade-off problem in project network analysis. *Foundations of Control Engineering*, *6*, 107–117.

Reda, R. M. (1990). RPM: repetitive project modelling. *Journal of Construction Engineering and Management*, *116*(2), 316–330.

Robinson, D. R. (1975). A dynamic programming solution to cost/time trade-off for CPM. *Management Science*, *22*, 158–166.

Schrage, L. (1995). *LINDO: Optimization software for linear programming*. Chicago: LINDO Systems.

Siemens, N. (1971). A simple CPM time/cost trade-off algorithm. *Management Science*, *17*, B-354–B-363.

Siemens, N., & Gooding, C. (1975). Reducing project duration at minimum cost: a time/cost trade-off algorithm. *OMEGA*, *3*, 569–581.

Skutella, M. (1998). Approximation algorithms for the discrete time-cost trade-off problem. *Mathematics of Operations Research*, *23*, 909–929.

Vanhoucke, M. (2005). New computational results for the discrete time/cost trade-off problem with time-switch constraints. *European Journal of Operational Research*, *165*, 359–374.

Vanhoucke, M. (2006). Work continuity constraints in project scheduling. *Journal of Construction Engineering and Management*, *132*, 14–25.

Vanhoucke, M., Demeulemeester, E., & Herroelen, W. (2001). On maximizing the net present value of a project under renewable resource constraints. *Management Science*, *47*, 1113–1121.

Vanhoucke, M., Demeulemeester, E., & Herroelen, W. (2002). Discrete time/cost trade-offs in project scheduling with time-switch constraints. *Journal of the Operational Research Society*, *53*, 741–751.

Vercellis, C. (1990). Multi-project scheduling with time-resource-cost trade-offs: a tactical model. In *Proceedings of the second international workshop on project management and scheduling*, Compiègne, France, 20–22 June 1990.

Wiest, J. D., & Levy, F. K. (1977). *A management guide to PERT/CPM: with GERT/PDM/DCPM and other networks*. New Jersey: Prentice-Hall.

Yang, H. H., & Chen, Y. L. (2000). Finding the critical path in an activity network with time-switch constraints. *European Journal of Operational Research*, *120*, 603–613.