Computers & Operations Research 37 (2010) 649–655

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/cor

Discrete time/cost trade-off problem: A decomposition-based solution algorithm for the budget version

Öncü Hazır[a,b,*], Mohamed Haouari[c,d], Erdal Erel[b]

[a] Industrial Engineering Department, Çankaya University, Ankara 06530, Turkey
[b] Faculty of Business Administration, Bilkent University, Ankara 06800, Turkey
[c] Department of Industrial and Systems Engineering, Faculty of Engineering, Ozyegin University, Istanbul, Turkey
[d] Princess Fatimah Alnijris's Research Chair for AMT, College of Engineering, King Saud University, Saudi Arabia

ARTICLE INFO

Available online 21 June 2009

Keywords:
Project scheduling
Time/cost trade-off
Benders Decomposition

ABSTRACT

This paper investigates the budget variant of the discrete time/cost trade-off problem (DTCTP). This multi-mode project scheduling problem requires assigning modes to the activities of a project so that the total completion time is minimized and the budget and the precedence constraints are satisfied. This problem is often encountered in practice as timely completion of the projects without exceeding the budget is crucial. The contribution of this paper to the literatures is to describe an effective Benders Decomposition-based exact algorithm to solve the DTCTP instances of realistic sizes. Although Benders Decomposition often exhibits a very slow convergence, we have included several algorithmic features to enhance the performance of the proposed tailored approach. Computational results attest to the efficacy of the proposed algorithm, which can solve large-scale instances to optimality.

1. Introduction

In project management, it is often possible to reduce the duration of some of the activities and therefore expedite the project duration with additional costs. This time/cost trade-off has been widely studied in the literatures. In this paper, we address a discrete version of the problem, namely the discrete time/cost trade-off problem (DTCTP).

Three versions of the DTCTP have been studied in the literatures so far: the deadline problem (DTCTP-D), the budget problem (DTCTP-B) and the efficiency problem (DTCTP-E). In DTCTP-D, given a set of time/cost pairs (modes) and a project deadline, each activity is assigned to one of the possible modes in such a way that the total cost is minimized. Conversely, the budget problem minimizes the project duration while not exceeding a given budget. On the other hand, DTCTP-E is the problem of constructing efficient time/cost solutions over the set of feasible project durations. This paper concentrates on the budget version. This problem is often encountered in practice as timely completion of the projects without exceeding the budget is crucial. Furthermore, in many projects, majority of the activities can be executed in various processing alternatives corresponding to different technologies or different resource assignments. Not surprisingly, the DTCTP has been shown to be strongly NP-hard for general activity networks [1].

The literature on the DTCTP is rather sparse. De et al. [2] review the problem and discuss some of the solution strategies. In order to solve the budget problem exactly, Robinson [3] and Demeulemeester et al. [4] proposed dynamic programming and branch and bound, respectively. These two studies solve small project networks with up to 45 activities. In our research, we aim to solve much larger problem instances (having more than 100 activities) optimally. Due to the computational complexity, Skutella [5] proposed an approximate algorithm for the budget problem. For the deadline version, we refer the readers to the studies of Hindelang and Muth [6], Demeulemeester et al. [7] for exact algorithms, and to Akkan et al. [8], and Vanhoucke and Debels [9] for approximate algorithms.

Formally, the DTCTP is defined as follows. A project is given with a set of n activities along with a corresponding precedence graph $G=(N, A)$, where N is the set of nodes, and $A \subset N \times N$ is the set of immediate precedence constraints on the activities. It is noteworthy that G also includes two dummy "start" and "end" nodes indexed by 0 and $n+1$, respectively. Each activity j $(j=1,\dots,n)$ can be performed at one of the $|M_j|$ modes, where M_j is the set of modes of activity j. Furthermore, each mode m is characterized by a processing time p_{jm} and a cost c_{jm}.

* Corresponding author at: Industrial Engineering Department, Çankaya University, Ankara 06530, Ankara, Turkey.
E-mail addresses: hazir@cankaya.edu.tr (O. Hazır), mohamed.haouari@ozyegin.edu.tr (M. Haouari), erel@bilkent.edu.tr (E. Erel).

0305-0548/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2009.06.009

A mixed integer linear programming formulation for the DTCTP-B can be constructed by defining for each activity j ($j = 1, \dots, n$) a continuous decision variable C_j which represents the completion time of activity j. Also, we define a binary decision variable x_{jm} ($j = 1, \dots, n$ and $m \in M_j$) that is equal to 1 if mode m is chosen for activity j, and 0 otherwise. The resulting formulation is given as follows:

$$\text{Minimize } C_{n+1} \tag{1.0}$$

subject to

$$\sum_{m \in M_j} x_{jm} = 1 \quad \forall j \in N \tag{1.1}$$

$$C_j - C_i - \sum_{m \in M_j} p_{jm} x_{jm} \geq 0 \quad \forall (i,j) \in A \tag{1.2}$$

$$\sum_{j \in N} \sum_{m \in M_j} c_{jm} x_{jm} \leq B \tag{1.3}$$

$$C_j \geq 0 \quad \forall j \in N \cup \{0, n+1\} \tag{1.4}$$

$$x_{jm} \in \{0,1\} \quad \forall m \in M_j,\ \forall j \in N \tag{1.5}$$

The objective function (1.0) is to minimize the project completion time. Constraints (1.1) require that a unique mode should be assigned to each activity. Constraints (1.2) are the precedence constraints. The knapsack constraint (1.3) requires that the budget, denoted with parameter B, should not be exceeded.

In this paper, we propose a tailored Benders Decomposition exact algorithm to solve the budget version of the deterministic DTCTP of realistic sizes. Benders [10] introduced this decomposition algorithm to solve specially structured large-scale linear and mixed integer programs. The basic idea is to decompose the problem into two simpler subproblems: the first part, called the master problem (MP), solves a relaxed version of the problem and generates trial values for the integer variables and a lower bound for a minimization objective. The second problem, called the subproblem (SP), is the original problem with the values of the integer variables temporarily fixed by the MP. The dual of the SP inserts cuts into the master problem and obtains an upper bound for a minimization objective. Up to now, Benders Decomposition has been used to solve many combinatorial optimization problems; specifically, successful applications of this methodology on network design are numerous [11]. However, only a few applications for project scheduling problems are reported in the literatures and these are discussed below.

Maniezzo and Mingozzi [12] use a heuristic algorithm based on Benders Decomposition to solve the Multi-Mode Resource Constrained Project Scheduling Problem (MRCPSP) approximately. In the MRCPSP, each activity is assigned to one of the possible modes and the starting time of each activity is determined so that the project completion time is minimized while precedence and resource constraints are satisfied. It is easy to see that the DTCTP-B is a special case of MRCPSP in which only a single nonrenewable resource (money) is in use, whereas the general MRCPSP allows the use of both renewable and nonrenewable resources. Maniezzo and Mingozzi [12] propose a mathematical formulation to the MRCPSP, relax some of the constraints and solve the relaxed version with a heuristic procedure based on Benders Decomposition. In their model, the MP corresponds to the assignment of one mode to each activity, while the SP is a single mode RCPSP. They solve both the MP and SP approximately.

Erengüç et al. [13] use Benders Decomposition to solve the time/cost trade-off problem with discounted cash flows, which is indeed a combination of the DTCTP and the payment-scheduling problem. Kuyumcu and Garcia-Diaz [14] solve the project compression problem with concave or convex piecewise linear cost-duration functions. At this point, it is worth mentioning that all these latter project scheduling applications were computationally assessed on small to medium size problem instances.

Our research differs from the studies which use Benders Decomposition in project scheduling regarding both the problem addressed and the solution approach. DTCTP-B has not been solved with Benders Decomposition. Furthermore, we propose some novel and problem acceleration mechanisms and integrate into the algorithm. Normally, Benders Decomposition is known to exhibit slow convergence and acceleration mechanisms are required [15,16]. We speed-up the solution process by using an enhanced algorithm, thus we render it capable of solving problems of realistic sizes. In the following sections, we present the details of the Benders Decomposition algorithm and enhancements for the DTCTP-B.

2. Benders Decomposition

2.1. Benders reformulation

Model (1.0)–(1.5) could be rewritten as follows:

$$\text{Min } C_{n+1}(x) \tag{2}$$

subject to (1.1), (1.3), and (1.5)where for each binary vector x, we have

$$C_{n+1}(x) \equiv \text{Min } C_{n+1} \tag{3.0}$$

subject to

$$C_j - C_i - \sum_{m \in M_j} p_{jm} x_{jm} \geq 0 \quad \forall (i,j) \in A \tag{3.1}$$

$$C_j \geq 0 \quad \forall j \in N \cup \{0, n+1\} \tag{3.2}$$

Using duality and defining w_{ij} as the dual variables associated with the constraint set (3.1), we rewrite $C_{n+1}(x)$ as

$$C_{n+1}(x) \equiv \text{Max} \sum_{(i,j) \in A} \left(\sum_{m \in M_j} p_{jm} x_{jm} \right) w_{ij} \tag{4.0}$$

subject to

$$\sum_{j:(j,i) \in A} w_{ji} - \sum_{j:(i,j) \in A} w_{ik} \leq 0 \quad \forall i \in N \cup \{0\} \tag{4.1}$$

$$\sum_{j:(j,n+1) \in A} w_{j,n+1} \leq 1 \tag{4.2}$$

$$w_{ij} \geq 0 \quad \forall (i,j) \in A \tag{4.3}$$

Therefore, it appears that given a binary vector x that is feasible (i.e. satisfying (1.1) and (1.3)), the computation of $C_{n+1}(x)$ requires solving a longest path in G between nodes 0 and n+1 with each edge $(i,j) \in A$ having a weight $\sum_{m \in M_j} p_{jm} x_{jm}$. For the sake of brevity, the formal proof is skipped.

Now, let $\overline{w^s}$ for $s = 1, \dots, S$, be the extreme points of the polyhedron defined by (4.1)–(4.3). Then, $C_{n+1}(x)$ can be stated as

$$C_{n+1}(x) \equiv \underset{s=1,\dots,S}{\text{Maximum}} \sum_{(i,j) \in A} \left(\sum_{m \in M_j} p_{jm} x_{jm} \right) \overline{w^s_{ij}} \tag{5}$$

or equivalently, $C_{n+1}(x)$ is the value of the optimal solution of the following LP:

$$\text{Min } z \tag{6.0}$$

subject to

$$z \geq \sum_{(i,j)\in A}\left(\sum_{m\in M_j} p_{jm}x_{jm}\right)\overline{w_{ij}^s} \quad s=1,\ldots,S \tag{6.1}$$

This yields the Benders reformulation of the problem defined by (1.0)–(1.5) as follows.

$$\text{Min}\, z \tag{7.0}$$

subject to

$$z \geq \sum_{(i,j)\in A}\sum_{m\in M_j}\overline{w_{ij}^s}p_{jm}x_{jm} \quad s=1,\ldots,S \tag{7.1}$$

$$\sum_{j\in N}\sum_{m\in M_j} c_{jm}x_{jm} \leq B \tag{7.2}$$

$$\sum_{m\in M_j} x_{jm}=1 \quad \forall j\in N \tag{7.3}$$

$$x_{jm}\in\{0,1\} \quad \forall m\in M_j,\ \forall j\in N \tag{7.4}$$

$$z\geq 0 \tag{7.5}$$

Note that, the constraint set (7.1) defines the paths of the network. However, enumerating all the paths is burdensome, therefore the paths are generated and the cuts (7.1) are appended as needed. Hence, our model is solved using a *relaxation approach*. At each iteration, feasibility is maintained and optimality cuts (7.1) are dynamically appended as outlined in the algorithm below.

In this algorithm, we represent the set of feasible mode assignments for the DTCTP-B by X^0, and introduce an index t to the notation to indicate the values at iteration t, such as x^t, X^t, MP^t, and z^t, refer to the solution, feasible set, relaxed master problem and the length of the longest path at time t, respectively. Furthermore, we will use the notation s^k to denote the kth longest path, hence s^1 will refer to a critical path. The following algorithm is an application of Benders Decomposition, introduced originally by Benders [10], to the DTCTP-B.

Benders Decomposition Algorithm for the DTCTP-B (*basic version*)

```
1. Start with an initial solution, $\overline{x^1}\in X^0$; set $LB=-\infty$, $UB=\infty$, $t=1$.
   Given $\overline{x^t}$, find out a critical path $s^1$, with length, $C_{n+1}(\overline{x^t})=\sum_{(i,j)\in A}\sum_{m\in M_j} w_{ij}^{s^1}p_{jm}\overline{x_{jm}^t}$Set $UB=\text{Min}\,\{UB,\ C_{n+1}(\overline{x^t})\}$If $(UB=LB)$
   Stop and report $\overline{x^t}$ as an optimal solution.
   Else
      $X^t=X^{t-1}\cap\{x\in X^0 : z\geq\sum_{(i,j)\in A}\sum_{m\in M_j} w_{ij}^{s^1}p_{jm}x_{jm}\}$
2. Solve the relaxed master problem, $MP^t$ : $z^t=\text{Min}\dot{}$Let $x^t$ be the optimal solution.
3. Set $LB=z^t$, $t=t+1$, $\overline{x^t}=x^{t-1}$.
4. Return to Step 2.
```

However, Benders Decomposition is known to converge slowly, so we propose some enhancements to accelerate the convergence and to solve large-scale instances to optimality.

2.2. Algorithmic enhancements

The main features of the enhanced approach are the following:

i. Preprocessing is integrated to eliminate some of the modes.
ii. Multiple cuts are inserted at each iteration.
iii. Approximate solutions of the relaxed master problems and LP relaxation are used to generate cuts.
iv. A local search improvement module is integrated to obtain a better feasible solution (hence, a tighter upper bound). Additional cuts are inserted from the improved solutions so that the global lower bound improves.
v. A customized branch-and-cut algorithm that groups the variables as special ordered sets (SOS) are developed to solve the MIP iterations.

2.2.1. Preprocessing

We integrate a preprocessing technique that serves to reduce the number of binary variables. At each iteration of Benders Algorithm, MP solves a relaxed problem and generates a LB. We invoke the preprocessing after each MP iteration. In the following theorem, we show that some of the long modes with long processing times could be redundant and hence could be eliminated.

Theorem. . *Given any LB for the DTCTP-B, for any activity i if there exists a mode m such that the length of the longest possible path that passes through activity i is less than or equal to the LB, then there exists an optimal solution such that activity i is performed either at mode m or at a mode with longer duration.*

Proof. Let $l_{uv}(x)$ be the length of the longest path between node u and v, and $A(x,i)$ be the length of the longest path that does not pass through activity i given the mode assignment, $x\in X^0$.

The longest possible path that passes through activity i is observed in the following mode assignment: Mode m is assigned to activity i and the modes with longest durations are assigned to the remaining activities. We will denote this assignment as x^L hereafter. Note that $x^L\in X^0$, otherwise as the modes with longest durations have the least cost, mode m would never become feasible and could be eliminated.

Let us assume that there exists an optimal solution x' such that $x_{\text{im}'}=1$, where $m<m'$; without loss of generality, for any $i\in N$, $m<m'$ implies $p_{\text{im}}>p_{\text{im}'}$ and $c_{\text{im}}<c_{\text{im}'}$.

$$\text{Given that, } p_{im}+l_{0i}(x^L)+l_{in+1}(x^L)\geq LB, \tag{8}$$

we have,

$$p_{im'}+l_{0i}(x')+l_{in}+1(x')p_{im}+l_{0i}(x^L)+l_{in}+1(x^L)\leq LB\leq C_{n+1}(x') \tag{9}$$

We keep all the mode assignments in x' other than activity i and assign mode m to activity and call this new assignment y, i.e. $y_{\text{im}}=1$ and $y_{\text{jm}}=x_{\text{jm}}\ \forall j\in Ni$.

Since $c_{\text{im}}<c_{\text{im}'}$, the mode assignment remains feasible, i.e. $y\in X^0$. For this new assignment,

$$A(y,i)=A(x',i)\leq C_{n+1}(x') \tag{10}$$

$$C_{n+1}(y)=\text{Max}\{p_{im}+l_{0i}(y)+l_{in+1}(y),A(y,i)\} \tag{11}$$

and

$$l_{0i}(y)+l_{in+1}(y)=l_{0i}(x')+l_{in+1}(x') \tag{12}$$

Using (10) and (12) we have

$$p_{im}+l_{0i}(y)+l_{in}+1(y)=p_{im}+l_{0i}(x')+l_{in+1}(x')\leq p_{im}+l_{0i}(x^L)+l_{in+1}(x^L)\leq LB\leq C_{n+1}(x'),$$

hence due to (10) and (11), $C_{n+1}(y)\leq C_{n+1}(x')$.

Therefore, when (8) holds for any optimal solution such that $x_{\text{im}'}=1$: $m<m'$, there exists an alternative optimal solution such that $x_{\text{im}''}=1$: $m''=m$ or $m''<m$. Hence, all the modes $m'>m$ could be eliminated. □

2.2.2. Multiple cuts

In our algorithm, we propose to find the longest K paths and insert multiple cuts at each iteration, instead of concentrating on the critical path and adding a single cut. In doing so, we observed in our computational study that the total number of iterations decreases drastically. For this purpose, we use a modification of the well-known Yen's [17] *K-shortest loopless paths* (KSP) algorithm and insert K cuts. As PERT networks are acyclic, critical path problems may be solved as shortest path problems with cost parameters equal to the negative of the activity durations.

In this strategy, the parameter K affects the computational efficiency; total number of iterations decreases with larger K, whereas total number of constraints increases. Considering this trade-off, we performed pretests involving problems with different sizes and set K to 30.

2.2.3. Approximate solutions

We solve the LP relaxation first and generate cuts from the fractional solutions. Next, integrality constraints are added and the algorithm is restarted [18]. Furthermore, not all of the master problems are solved to optimality, i.e. feasibility cuts are generated from approximate solutions; the branch-and-bound algorithm is truncated. The algorithm starts with an initial relative optimality tolerance level ε (in our implementation, we set $\varepsilon = 4\%$) and we use an *adaptive strategy* which monitors the optimality gap, $100(UB-LB/LB)$. If the optimality gap value is not improved for three consecutive iterations, then the tolerance gap is halved (that is, we set $\varepsilon = \varepsilon/2$). This works to improve the lower bound. Note that the truncated solution of the relaxed MP does not necessarily provide a lower bound, however considering the tolerance level, if F_R is the value of the ε-optimal solution, then it is easily realized that $LB = F_R/1+\varepsilon$ is a valid lower bound.

2.2.4. Local search

After solving the relaxed master problem, we invoke a local search procedure that aims at producing an improved solution and therefore accelerating convergence. More precisely, starting from an initial solution, the procedure generates and moves to neighbor solutions with the following strategy: First using the initial solution, *potentially critical activities* (*PCA*) are identified. The conventional measure of an activity criticality is its total slack, which is the amount of time by which the completion time of an activity can exceed its earliest completion time without delaying the project completion time. In the literatures, the activities that have zero slacks are defined to be critical activities. We enlarge the set of critical activities and define the activities that have total slacks less than $100\xi\%$ of the activity duration as potentially critical activities, i.e., $PCA = \{j\text{: } TS_j/p_j \le \xi\}$. TS_j is the total slack of activity *i*. In our implementation, we set $\xi = 0.15$. Having defined the PCA, we fix the modes of the activities which do not belong to PCA, and solve a reduced instance that decides on the modes of only potentially critical activities. As reduced instances contain much fewer decision variables, it is possible to solve them directly with a commercial general-purpose solver (in our implementation, we used CPLEX 9.1). The exact algorithm is based on the formulation given through Eqs. (1.0)–(1.5). However, in order to speed-up the solution further, the branch and bound is stopped if the problem is solved to within 2% of optimality, i.e., $\varepsilon = 2\%$. Local search is reiterated until convergence.

As a supplementary strategy to improve the value of the lower bound, we propose to generate additional cuts derived from the improved solution obtained with the local search. Therefore, we could generate $K = K_1+K_2$, cuts derived from the initial and improved solutions. Having performed several pretests, we set $K_1 = 10$, $K_2 = 20$.

2.2.5. Branch and cut

In a classical branch-and-bound algorithm, for each node the LP relaxation is solved, and a fractional variable (if there is one) on which to branch is chosen. In order to solve the MP efficiently, we branch on sets of variables instead of branching on individual variables. A set of variables, in which at most one variable in the set is allowed to be nonzero, forms a "Special Ordered Set of Type 1" (SOS_1). In the DTCTP, the constraint set (1.1) represents an SOS_1. Branching strategies have large impacts on the size of the branch-and-bound tree and the computation time. Branching on SOS_1 instead of a single variable has some advantages such that the tree becomes more balanced [19].

We assign an order to the variables among SOS_1 by using the activity durations as weights. The variables are divided into two sets: the first subset includes the variables that have weights greater than the average weight, as calculated by using the solution of the relaxed problem. The second subset consists of variables that have weights less than the average value. Two branches are created by setting the variables in each subset to zero. We develop the branching tree using CPLEX and select the branching nodes using a best-estimate search strategy, a strategy in which one chooses the node estimated to have the best feasible integer solution obtainable.

Integrating all the proposed enhancements, we propose the following modified Benders Decomposition Algorithm to solve the DTCTP-B.

Benders Decomposition Algorithm for the DTCTP-B (*Enhanced Version*)

```
1. Start with an initial solution, $\overline{x^1} \in X^0$; set $LB = -\infty$, $UB = \infty$, $t = 1$.
   Given $\overline{x^t}$, find out a critical path $s^1$, and its length, $C_{n+1}(\overline{x^t}) = \sum_{(i,j)\in A}\sum_{m\in M_j} w_{ij}^{s^1} p_{jm}\overline{x_{jm}^t}$ Set $UB = Min\ \{UB,\ C_{n+1}(\overline{x^t})\}$ If ($UB = LB$)
   Stop and report $\overline{x^t}$ as the optimal solution.
   Else
   $X^t = X^{t-1} \cap \{x \in X^0 : z \ge \sum_{(i,j)\in A}\sum_{m\in M_j} w_{ij}^{s^k} p_{jm}\overline{x_{jm}^t}\}$, $k = 1,\dots,K_1$ and set Improved $= 0$
2. Invoke Local Search. Let $\tilde{x}^t$ be the solution obtained by the Local Search.
   If $C_{n+1}(\tilde{x}^t) < C_{n+1}(\overline{x^t})$ then set Improved $= 1$, set $\overline{x^t} = \tilde{x}^t$ and return to Step 3.
3. If Improved $= 1$ then
   Set $\{UB = \text{Min } C_{n+1}(\tilde{x}^t)\}$ If ($UB = LB$)
   Stop and report $\tilde{x}^t$ as the optimal solution.
   Else
   $X^t = X^{t-1} \cap \{x \in X^0 : z \ge \sum_{(i,j)\in A}\sum_{m\in M_j} w_{ij}^{s^k} p_{jm}\overline{x_{jm}^t}\}$, $k = 1,\dots,K_2$.
4. Solve the relaxed master problem, $MP^t$: $z^t = \text{Min}\ \{z\text{: } x \in X^t\}$. Let $x^t$ be the solution.
5. Set $LB = z^t$, $t = t+1$, $\overline{x^t} = x^{t-1}$.
6. Invoke Preprocessing.
7. Return to Step 2
```

3. Experimentation and computational results

We performed computational experiments to measure the efficiency of the algorithm under various problem settings. For experimentation, we generate 240 problem instances, corresponding to 48 project settings with 5 replicates, from the project networks provided by Akkan et al. [8]. We use the test-bed 1, which includes large sized projects having 85–136 activities.

Network structure of a project is defined mainly with two parameters: complexity index (CI) and the coefficient of network complexity (CNC). CI is a measure developed by Bein et al. [20] to assess how far the given network is from being series–parallel. It is defined to be the minimum number of node reductions required to reduce

Table 1
Experimental setting.

Parameters	Level(s)
CI	13,14
CNC	5, 6, 7, 8
# Modes	U[2,10], U[11,20]
Budget parameter (θ)	0.15
Cost function (CF)	ccv, cvx, hyb

Table 2
Summary of computational results (# modes $\in$ U[2,10]).

Cost function	CI	CNC	LP Iter	IP Iter	Mode elim (%)	Avg CPU(s)	Max CPU(s)
CCV	13	5	17.80	4.40	15.47	30.66	95.20
		6	22.60	4.80	14.92	413.47	1703.10
		7	27.80	19.40	14.37	768.07	1317.19
		8	30.20	16.80	10.82	1524.25	3879.23
	14	5	11.40	4.00	17.82	30.93	129.55
		6	15.40	8.60	12.64	126.90	452.62
		7	25.20	15.80	12.66	1014.74	2054.28
		8	27.40	19.80	12.00	1743.32	4051.49
CVX	13	5	16.60	6.20	15.22	42.68	97.22
		6	23.00	12.20	12.87	435.04	1407.64
		7	22.80	13.40	11.59	821.42	2430.85
		8	28.00	22.20	10.04	1407.25	3010.26
	14	5	11.80	5.60	13.36	23.81	50.68
		6	15.60	7.20	8.87	110.87	439.84
		7	23.60	15.80	9.42	987.32	2031.33
		8	28.50	18.25	8.91	1867.62	4432.03
HYB	13	5	16.00	5.40	18.36	49.57	171.53
		6	21.00	7.00	12.85	199.21	379.39
		7	29.20	12.20	13.93	906.82	3518.43
		8	26.40	14.60	11.04	850.52	1424.29
	14	5	12.20	3.60	12.76	12.95	22.93
		6	16.40	9.00	11.13	143.39	568.95
		7	24.20	15.60	11.70	355.33	681.89
		8	30.00	20.80	10.89	1191.06	4058.48

a given two terminal directed acyclic graph into a single-arc graph, when used together with series and parallel reductions. Assessing the distance of a given network from being series–parallel is important for this study, because DTCTP with series–parallel graphs could be solved quickly [4]. The second complexity measure, CNC is developed by Pascoe [21] and defined to be the ratio of the number of arcs to the number of nodes.

The number of modes per activity is randomly generated with discrete uniform distribution using intervals U[2,10] and U[11,20]. To compute the budget parameter for each instance, first the minimum possible project cost, $C_{\min}$ (total cost with cheapest modes) and the maximum possible project cost, $C_{\max}$ (total cost with most expensive modes), are calculated. Then, the budget is set as follows:

$$B_0 = C_{\min} + \theta(C_{\max} - C_{\min}), \quad \text{where } \theta = 0.15 \tag{13}$$

Concave (ccv), convex (cvx), and neither concave nor convex cost functions (hyb) are used to generate the cost figures. Table 1 summarizes the parameters of the test bed.

In our experimentation, we categorize the instances with respect to the number of modes and with respect to the CNC. These parameters define total processing alternatives and total number of activities; therefore, they set the binary decision variables and affect the computational complexity.

First, we solve the 120 large projects with moderate number of modes, with 85–136 activities and the number of modes lies in the interval U[2,10]. Computational results of this set are reported in Table 2. According to the results, 74.17% (89/120) of these instances could be solved exactly in 10 min, 95.83% (115/120) in an hour and all the instances within 90 min. In order to represent the complexity of real life projects, our test instances are far beyond the problem sizes reported by Demeulemeester et al. [4]. Problems with similar sizes could be solved within only a few seconds with our algorithm.

Secondly, we solve 120 hard instances having a number of modes lying in the interval U[11,20] So as to assess the efficiency of the proposed algorithm, we also solve the MIP formulation given in 1.0–1.5 with optimization software CPLEX 9.1 and compare the results. A maximal time limit of 3 h is set for each exact procedure. We report the computational results with respect to the coefficient of network complexity (CNC), which is the ratio of the number of arcs to the number of nodes. Tables 3 and 4 summarize the experimental results for the instances with $CNC = \{5, 6\}$ and $CNC = \{7, 8\}$, respectively.

In Table 3, we present the results of the Benders Decomposition-based exact procedure and CPLEX implementation under the columns labeled with “BENDERS DECOMPOSITION” and “CPLEX”, respectively; the percentage of problem instances that the optimal solution is found within the time limit and the average CPU time of the problem instances for which an optimal solution is found with Benders algorithm are reported under the columns “Ins Opt (%)” and “CPU(s)”, respectively. CPLEX CPU time is assumed to be 3 h for instances that CPLEX could not solve within the time limit, hence average CPLEX CPU times are underestimated. Additionally, we report the percentage of eliminated modes, average and maximum gaps (only the problem instances for which an optimal solution is found within the time limit) of the Benders Decomposition-based exact procedure under the columns “Mode Elim (%)”, “Avg Gap (%)”, and “Max Gap (%)”. The gap corresponds to the relative difference between the best lower and upper bounds.

Interestingly, we observe that a great majority of the instances with $CNC = 5$, 6 (projects with 85–106 activities) could be solved exactly within the time limits with our algorithm, i.e. 88.33% (53/60), whereas 61.67% (37/60) for CPLEX. The seven unsolved instances exhibit an average gap of only 0.91%. Moreover, when an instance is solved by both, Benders Decomposition requires shorter CPU times. On the other hand, when $CNC = 7$, 8; we found that the algorithm fails to produce proven optimal solutions. However, the algorithm generates solutions that are close to the optimal solution; the average and maximum gaps between upper and lower bounds are 1.31% and 2.36%, respectively. Table 4 also illustrates the impact of integrating the local search on the performance of the algorithms.

A remarkable result is that integrating the local search decreases the gaps between upper and lower bounds significantly. We also investigated the overall effect of all the enhancements presented in Section 2.2 and found out that the improvements are very drastic. For example, one of the smallest instances (with $CI = 13$, $CNC = 5$, $n = 85$) which Benders could solve in 58 s, classical approach required 2415 s.

We applied Fixed Effects ANOVA test to the results in order to find out the variance effect of the experimental design factors to CPU time (only for $CNC = 5$,6; for these cases a great majority of the instances could be solved exactly within time limits) and to the number of modes eliminated (for all instances). The treatment levels are fixed for these following factors: CNC, CI, and type of cost function (CF). Furthermore, the interactions between these factors are investigated. Before the test is performed, the necessary assumptions for ANOVA are checked and Box–Cox transformation is utilized to restore constant error variances and normality assumption. Test results are reported in Tables 5 and 6.

We also illustrate the relationship between design factors to CPU time and to the number of modes eliminated in Figs. 1 and 2, respectively.

Table 3
Summary of computational results (CNC $\in$ {5, 6}; # modes $\in$ U[11,20]).

Cost function	CI	CNC	CPLEX		BENDERS DECOMPOSITION				
			Ins opt (%)	CPU(s)	Ins opt (%)	Mode elim (%)	CPU(s)	Avg gap (%)	Max gap (%)
CCV	13	5	100.00	984.61	100.00	14.88	465.21	–	–
		6	60.00	3557.72	80.00	12.20	3020.66	1.87	1.87
	14	5	100.00	96.86	100.00	11.53	138.71	–	–
		6	100.00	1334.25	100.00	9.48	706.17	–	–
CVX	13	5	20.00	9281.64	100.00	9.24	1590.69	–	–
		6	0.00	–	0.00	7.04	–	0.72	1.00
	14	5	80.00	6288.21	100.00	7.78	1498.94	–	–
		6	0.00	10800.00	80.00	5.21	4833.99	0.89	0.89
HYB	13	5	80.00	3680.55	100.00	14.47	283.88	–	–
		6	60.00	5691.62	100.00	10.77	619.09	–	–
	14	5	100.00	1409.49	100.00	10.46	244.71	–	–
		6	40.00	6663.28	100.00	7.31	1062.73	–	–

Table 4
Summary of computational results (CNC $\in$ {7,8}; # Modes $\in$ U[11,20]).

Cost function	CI	CNC	BENDERS DECOMPOSITION				
			Mode elim (%)	Without local search		With local search	
				Avg gap (%)	Max gap (%)	Avg gap (%)	Max gap (%)
CCV	13	7	9.94	2.32	3.39	1.71	2.36
		8	8.05	3.11	3.2	2.12	2.12
	14	7	8.21	2.56	3.29	1.51	2.30
		8	8.60	2.09	3.44	1.27	1.58
CVX	13	7	5.80	1.64	2.61	0.99	1.60
		8	4.62	1.99	3.20	1.21	1.50
	14	7	4.18	2.05	2.44	1.59	1.88
		8	4.37	2.47	3.22	1.72	1.97
HYB	13	7	8.89	0.95	2.19	1.16	1.78
		8	6.42	0.78	1.58	1.08	1.71
	14	7	5.99	1.05	1.56	0.92	1.38
		8	6.71	1.38	2.27	1.03	1.47

Table 5
Effect of factors on the CPU time: ANOVA test.

Source	DF	Sum of squares	Mean square	F	p
CI	1	0.180	0.180	2.282	0.137
CNC	1	2.420	2.420	30.628	0.000
CF	2	3.611	1.805	22.848	0.000
CI∗CNC	1	0.060	0.060	0.758	0.388
CI∗CF	2	0.070	0.035	0.446	0.643
CNC∗CF	2	0.272	0.136	1.718	0.190
Error	50	3.952	0.079		
Total	59	10.565	$\mathbf{R^2 = 0.626}$		

Table 6
Effect of factors on number of modes eliminated: ANOVA test.

Source	DF	Sum of squares	Mean square	F	p
CI	1	105.319	105.319	34.183	0.000
CNC	3	429.082	143.027	46.422	0.000
CF	2	388.200	194.100	62.999	0.000
CI∗CNC	3	45.671	15.224	4.941	0.003
CI∗CF	2	7.618	3.809	1.236	0.295
CNC∗CF	6	11.164	1.861	0.604	0.727
Error	102	314.274	3.081		
Total	119	1301.329	$\mathbf{R^2 = 0.759}$		

Network complexity is mainly measured with two measures: CI and CNC. We have not observed a significant effect of CI on computational effort in our experiments that is a result in line with Akkan et al.'s [8] finding for approximate solutions of DTCTP-D. However, it is easier to eliminate the modes of networks that have structures close to being series–parallel; hence CI is significantly effective on the performance of the preprocessing method. On the other hand, as CNC increases, the number of binary variables increases as well. Hence, the larger the CNC, the larger is the problem complexity. Furthermore, there exists an interaction between network complexity parameters.

The type of cost function is influential on both the computational effort and also on the effectiveness of the preprocessing method. The convex cost functions usually demand more computational effort and this is mainly due to the fact that less of the modes could be eliminated with preprocessing for these types of problem instances. This is illustrated in Fig. 2.

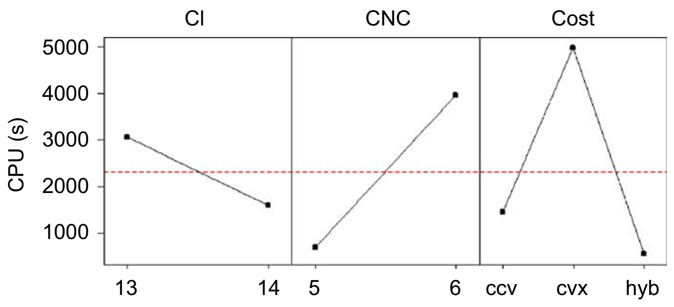

Fig. 1. Effect of design factors on CPU time: main effects plot.

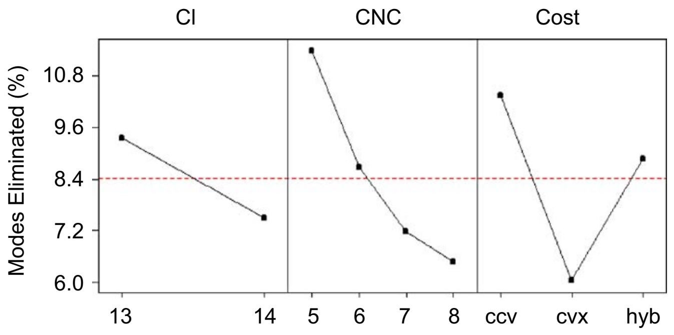

Fig. 2. Effect of design factors on mode elimination: main effects plot.

4. Conclusions

The DTCTP is a well-known project scheduling problem with practical implications. In this paper, we have mainly investigated the budget version and have proposed an exact algorithm to solve the problems of realistic sizes. The major contribution of this paper lies in the decomposition approach and the developed branch-and-cut procedure. We have included several features to accelerate the convergence into the solution algorithm and in this way; we manage to solve large instances, project networks with up to 136 activities to optimality. The scheduling algorithm presented in this paper could support project managers in scheduling large-scale projects with limited budgets.

Computational experiments are performed to measure the efficiency of the algorithm under various problem settings. The results have been compared with the ones obtained with state-of-the-art optimization software, CPLEX. We have experimentally shown that the proposed enhanced approach consistently outperforms both the basic Benders algorithm and a state-of-the-art commercial solver.

Mainly three factors are affecting the difficulty in solving a particular problem instance: the network structure, the number of modes per activity and the tightness of the budget. The major advantage of the proposed algorithm is that the optimal solutions can be obtained for projects with complex network structures, large number of modes and tight budgets. Moreover, near-optimal solutions could be derived for the hardest large-scale instances.

It would be of interest to investigate the budget version of the Multi-Mode Resource Constrained Project Scheduling Problem (MRCPSP), in which both renewable and nonrenewable resources are controlled simultaneously. We believe that the algorithms developed in this paper might prove as a useful base for investigating this challenging problem.

Acknowledgements

Dr. Mohamed Haouari would like to thank Fatimah Alnijris' Research Chair for Advanced Manufacturing Technology for the financial support provided for this research.

References

[1] De P, Dunne EJ, Ghosh JB, Wells CE. Complexity of the discrete time/cost trade-off problem for project networks. Operations Research 1997;45:302–6.
[2] De P, Dunne EJ, Ghosh JB, Wells CE. The discrete time/cost trade-off problem revisited. European Journal of Operational Research 1995;81:225–38.
[3] Robinson DR. A dynamic programming solution to the cost–time trade off for CPM. Management Science 1975;22:158–66.
[4] Demeulemeester E, Herroelen W, Elmaghraby SE. Optimal procedures for the discrete time/cost trade-off problem in project networks. European Journal of Operational Research 1996;88:50–68.
[5] Skutella M. Approximation algorithms for the discrete time–cost trade-off problem. Mathematics of Operations Research 1998;23:195–203.
[6] Hindelang TJ, Muth JF. Dynamic programming algorithm for decision CPM networks. Operations Research 1979;27:225–41.
[7] Demeulemeester E, De Reyck B, Foubert B, Herroelen W, Vanhoucke M. New computational results for the discrete time/cost trade-off problem in project networks. Journal of the Operational Research Society 1998;49:1153–63.
[8] Akkan C, Drexl A, Kimms A. Network decomposition-based benchmark results for the discrete time–cost trade-off problem. European Journal of Operational Research 2005;165:339–58.
[9] Vanhoucke M, Debels D. The discrete time/cost trade-off problem under various assumptions exact and heuristic procedures. Journal of Scheduling 2007; 10:311–26.
[10] Benders JF. Partitioning procedures for solving mixed variables programming problems. Numerische Mathematic 1962;4:238–52.
[11] Costa AM. A survey on benders decomposition applied to fixed charge network design problems. Computers and Operations Research 2005;32:1429–50.
[12] Maniezzo V, Mingozzi A. A Heuriic Ocedure for the multi-mode project scheduling problem based on Bender's decomposition. In: Weglarz J, editor. Project scheduling-—recent models, algorithms and applications. Boston: Kluwer Academic Publishers; 1999. p. 179–96.
[13] Erenguc SS, Tufekci S, Zappe CJ. Solving time/cost trade-off problems with discounted cash flows using generalized benders decomposition. Naval Research Logistics Quarterly 1993;40:25–50.
[14] Kuyumcu A, Garcia-Diaz A. A decomposition approach to project compression with concave activity cost functions. IIE Transactions 1994;26(6):63–73.
[15] Magnanti TL, Wong RT. Accelerating benders decomposition algorithmic enhancement and model selection criteria. Operations Research 1981;29: 464–84.
[16] Rei W, Cordeau JF, Gendreau M, Soriano P. Accelerating benders decomposition by local branching. INFORMS Journal on Computing 2007;19:534–41.
[17] Yen JY. Finding the K shortest loopless paths in a network. Management Science 1971;17:712–6.
[18] McDaniel D, Devine M. A modified Benders' partitioning algorithm for mixed integer programming. Management Science 1977;24:312–79.
[19] Linderoth J, Savelsbergh MWP. A computational study of search strategies for mixed integer programming. INFORMS Journal on Computing 1999;11:173–87.
[20] Bein WW, Kamburowski J, Stallmann MFM. Optimal reduction of two-terminal directed acyclic graphs. SIAM Journal on Computing 1992;21:1112–29.
[21] Pascoe TL. Allocation of resources–CPM. Revue Française de Recherche Opérationelle 1966;38:31–8.