PRZEGLĄD STATYSTYCZNY
R. LX – ZESZYT 3 – 2013

MARCIN ANHOLCER, HELENA GASPARS-WIELOCH

ACCURACY OF THE KAUFMANN AND DESBAZEILLE ALGORITHM FOR TIME-COST TRADE-OFF PROJECT PROBLEMS

1. INTRODUCTION

The Kaufmann and Desbazeille algorithm is a procedure used in the time-cost trade-off project analysis (TCTP-analysis). The method in question is commonly known by project management trainers, academic teachers and students. The algorithm is presented, among other places, in Bladowski (1970), Gaspars (2006a), Gaspars (2006b), Hendrickson (1989), Idźkiewicz (1967), Kukuła et al. (1996), Muller (1965), Muller (1964), Siudak (1998) and Trocki et al. (2003). Authors use different names for this method (e.g. CPM-COST analysis, time-cost analysis, network compression algorithm, MCX – Minimum Cost Expediting). That is why Kaufmann and Desbazeille (1964) are little-known surnames in TCTP analysis. Nevertheless, their book is the oldest one containing a description of the algorithm under consideration. Therefore, the procedure analyzed in this article, is conventionally called the Kaufmann and Desbazeille algorithm. For convenience, we also use of the abbreviation KDA.

Many people claim that this method is an exact algorithm (Bladowski, 1970; Bozarth, Handfield, 2005; Fusek et al., 1967; Giard, 1991; Idźkiewicz, 1967; Kopańska-Bródka, 1998; Kukuła et al., 1996; Muller, 1965; Trocki et al., 2003; Waters, 1998), i.e. an algorithm which always indicates the cheapest way of project compression. The KDA is however only a heuristic procedure. Examples of problems for which the method leads to quasi-optimal solutions are given in Gaspars (2006a) and Gaspars (2006b).

In Anholcer, Gaspars-Wieloch (2011) we have discussed and proved the special case for which the Kaufmann and Desbazeille algorithm gives the worst results. In this paper we are going to calculate the average accuracy of the KDA for several test problems similar to the worst case and for some randomly generated problems, as well.

2. TIME-COST TRADE-OFF PROJECT PROBLEMS

In TCTP problems the trade-off occurs between the project completion time (T) and the amount of non-renewable resources, i.e. money, which constitute the total cost of the project (TC).

The total cost (*TC*) consists of direct (*DC*) and indirect costs (*IC*). The first category concerns costs related directly to the completion of activities (e.g. labor, raw materials) and their compression. Activity durations are bounded from below (crash duration) and from above (normal duration). The shortening of a given task requires the increase of the amount of resources that are used to accomplish this activity. Indirect costs, like e.g. penalty, insurance and taxes, are assigned to the project as a whole. The penalty costs are imposed when the completion time has been delayed. In general, when the completion time increases, indirect costs increase and direct costs decrease (see Figure 1).

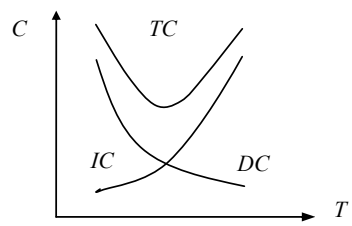

Figure 1. Time-cost curves

The main goals considered in TCTP analysis are as follows (Skutella, 1998):

1) minimizing time-dependent project costs within a specified project deadline or target time T^d (Deadline Problem),
$$C(X) \to \min, \tag{1}$$
$$T(X) \leq T^d, \tag{2}$$

2) minimizing project completion time within a specified budget C^b (Budget Problem),
$$T(X) \to \min, \tag{3}$$
$$C(X) \leq C^b, \tag{4}$$

where X signifies the vector of activity durations.

3. EXISTING ALGORITHMS FOR THE TCTP PROBLEM

A detailed survey and analysis of optimization methods applicable to the TCTP problems was presented by one of us in Gaspars-Wieloch (2008) and Gaspars-Wieloch (2009). In addition, we both showed a comparative breakdown of these algorithms in Anholcer, Gaspars-Wieloch (2011). Here we just recapitulate the most important parts of the breakdown mentioned above (t^c_i – the shortest possible time for the i-th activity, i.e. crash time; t^n_i – the normal time for the i-th activity), see Table 1.

Table 1.

Comparative breakdown of existing project time-cost trade-off algorithms

Criterion				Authors of procedures
1.	Type of optimization problems solved	$C(X) \rightarrow \min$, $T(X) \leq T^d$		1. Berman 2. Bladowski 3. De, Dunne, Gosh and Wells 4.Falk and Horowitz 5. Fondhal 6. Fulkerson 7. Gedymin 8. Goyal 9. Hindelang and Muth 10. Kaufmann and Desbazeille 11. Kelley 12. Liu, Burns and Feng 13. Moder and Phillips 14. Moussourakis and Haksever 15. Panagiotakopoulos 16. Phillips and Dessouky 17. Prager 18. Siemens
		$T(X) \rightarrow \min$, $C(X) \leq C^d$		1. Bladowski 2. De, Dunne, Gosh and Wells 3. Fulkerson 4. Gedymin 5. Kaufmann and Desbazeille 6. Kelley 7. Liu, Burns and Feng 8. Moder and Phillips 9. Moussourakis and Haksever 10. Phillips and Dessouky 11. Prager
		$C(T^{ad}) \rightarrow \min$ *		1. Bladowski 2. Crowston and Thompson 3. De, Dunne, Gosh and Wells 4. Fulkerson 5. Gedymin 6. Hindelang and Muth 7. Kaufmann and Desbazeille 8. Kelley 9. Liu Burns and Feng 10. Moder and Phillips 11. Moussourakis and Haksever 12. Phillips and Dessouky 13. Prager
2.	Type of costs considered	Shortening costs		1. Bladowski 2. Fulkerson 3. Gedymin 4. Goyal 5. Kaufmann and Desbazeille 6. Kelley 7. Panagiotakopoulos 8. Phillips and Dessouky 9. Prager 10. Siemens
		Shortening costs and other direct project costs		1. Berman 2. Bladowski 3. De, Dunne, Gosh and Wells 4. Falk and Horowitz 5. Fondhal 6. Kaufmann and Desbazeille 7. Liu, Burns and Feng 8. Moussourakis and Haksever
		All costs (direct and indirect)		1. Crowston and Thompson 2. Hindelang and Muth 3. Moder and Phillips
3.	Growth speed of direct costs	Constant (linear time-cost curve)		1. Fulkerson 2. Goyal 3. Kelley 4. Phillips and Dessouky 5. Prager 6. Siemens
		Increasing (convex curve)	Linear approximation	1. Kelley 2. Goyal 3. Liu, Burns and Feng 4. Phillips and Dessouky 5. Prager 6. Siemens
			No approximation	1. Berman
		Decreasing (concave curve)	Linear approximation	1. Falk and Horowitz 2. Gedymin
			No approximation	–
		Various (ex. Concave-convex curve)	Linear approximation	1. Falk and Horowitz 2. Gedymin 3. Moussourakis and Haksever
			No approximation	1. Bladowski 2. Crowston and Thompson 3. De, Dunne, Gosh and Wells 4. Fondhal 5. Hindelang and Muth 6. Kaufmann and Desbazeille 7. Moder and Phillips 8. Panagiotakopoulos

Table 1.

Criterion			Authors of procedures
4.	Feasible activity durations	Any real number from $<t^c_i, t^n_i>$	1. Berman 2. Falk and Horowitz 3. Fondhal 4. Fulkerson 5. Gedymin 6. Goyal 7. Kelley 8. Phillips and Dessouky 9. Prager 10. Siemens
		Any integer from $<t^c_i, t^n_i>$	1. Bladowski 2. Kaufmann and Desbazeille
		Discrete options from $<t^c_i, t^n_i>$	1. Crowston and Thompson 2. De, Dunne, Gosh and Wells 3. Hindelang and Muth 4. Liu, Burns and Feng 5. Moder and Phillips 6. Moussourakis and Haksever 7. Panagiotakopoulos
5.	Accuracy of solutions obtained	Optimal solutions	1. Berman 2. De, Dunne, Gosh and Wells 3. Falk and Horowitz 4. Fulkerson 5. Gedymin 6. Kelley 7. Liu, Burns and Feng 8. Moussourakis and Haksever 9. Phillips and Dessouky 10. Prager
		Optimal or suboptimal solutions	1. Bladowski 2. Crowston and Thompson 3. Fondhal 4. Goyal 5. Hindelang and Muth 6. Kaufmann and Desbazeille 7. Moder and Phillips 8. Panagiotakopoulos 9. Siemens

* T^{ad} denotes an advisable, but not mandatory, project completion time. When the advisable time is exceeded, than the total project cost must include a penalty. When the project is finished before the advisable time, the total cost is reduced by a bonus.

Source: Anholcer, Gaspars-Wieloch (2011), Gaspars-Wieloch (2008), Gaspars-Wieloch (2009).

Exact algorithms (i.e. those which guarantee finding an optimal solution) applied in discrete problems (*DTCTP* – Discrete Time-Cost Trade-off Problems) are characterized by an exponential complexity and are NP-hard. Procedures designed for continuous cases are solvable in polynomial time (see De et al., 1995; De et al., 1997; Gaspars-Wieloch, 2008; Panagiotakopoulos, 1977; Siudak, 1998; Skutella, 1998).

4. THE PROJECT NETWORK – NOTATION

In the paper we use the AOA (activities on arcs) network representation of the project, similarly as in Anholcer, Gaspars-Wieloch (2011). To be more exact, the set E of arcs consists of m arcs $e_1, \ldots, e_m$, while the set of nodes V consists of n nodes $v_1, \ldots, v_n$. Each arc e_i, $i = 1, \ldots, m$ is labeled by some positive natural number t_i, i.e. the duration of the respective activity. For convenience, we often index the arc by e_{jk}, where $j = 1, \ldots, n-1$ and $k = 2, \ldots, n$ are the indices of the starting and end node of the arc, respectively.

In addition, for each arc we define a non-decreasing sequence $C^{(i)} = \left(c^{(i)}_{k_i}\right)_{k_i=1}^{K_i}$ of real numbers representing the shortening cost, where $c^{(i)}_{k_i}$ is the cost of reducing the

duration of the i-th activity by the k_i-th unit and $K_i \leq t_i$ (one may shorten an activity at most $(t^n_i - t^c_i)$ times).

The earliest time of the event j (the earliest time at which node j can be reached such that all its preceding activities have been finished) is denoted by t^I_j (t^I_n being equal to the minimum completion time of the project, i.e. T^*). The latest time of the event j (the latest time that node j can be left such that it is still possible to finish the overall project in the minimum completion time) is denoted by t^{II}_j.

5. THE KAUFMANN AND DESBAZEILLE ALGORITHM – DESCRIPTION AND IMPLEMENTATION

The Kaufmann and Desbazeille algorithm is one of the oldest procedures applied in TCTP analysis. As one can notice in the Table 1, the KDA allows to solve the problem (1)-(2) or (3)-(4). This procedure focuses on the minimization of direct costs (see Section 2). Theoretically, the algorithm may be used in time-cost trade-off problems with any types of time-cost curves for project activities. However, in practice it is applied only when the unit shortening costs are non-decreasing, because in other cases the solutions obtained are extremely bad (i.e. far from the optimal ones). Most of the existing exact algorithms for TCTP are designed just for linear or convex time-cost curves.

Kaufmann and Desbazeille assume that the parameters representing the normal activity durations (t_i^n), the project target time (T^d) and the difference (t_i^n-t_i^c) are integers. They take into consideration only integer realizations of the project time even if, for a given C^b (see the Budget Problem (3)-(4)), the optimal solution requires non-integer times for some tasks (see Table 1, point 4).

The original version of the KDA consists in iteratively shortening each critical path in the network by exactly one time unit. The target is to find the cheapest way of compressing the project time by one unit. We stop when constraints (1)-(2) or (3)-(4) are satisfied. In order to apply this procedure, the user should know:

– the structure of the project network,
– the normal and the crash duration of the activities within the project,
– the unit shortening cost for each task,
– the deadline or budgetary constraint.

Let us recall the details of the KDA. Two steps are performed in every iteration: one is to implement the CPM method for the actual durations of activities and the second one is the shortening of the project by one time unit. The algorithm stops when the desired time (T^d) has been reached.

To compress the project duration, it is necessary to reduce each critical path by one time unit. It is not necessary to shorten one activity from each critical path as the critical paths not always are disjoint. In order to reduce the project duration, all the cuts of the critical sub-network are considered and the cheapest one is chosen. Then,

all the activities belonging to the minimal cut are shortened by one time unit. In the original KDA the cut P is defined as the set of arcs such that:

1. After removing all the arcs belonging to P, the project network is no longer connected.
2. The set of nodes V splits into two disjoint subsets: V_1 containing the starting event and V_2 containing the end event of the project such that the starting nodes of all the arcs from P belong to V_1, while the end nodes belong to V_2.

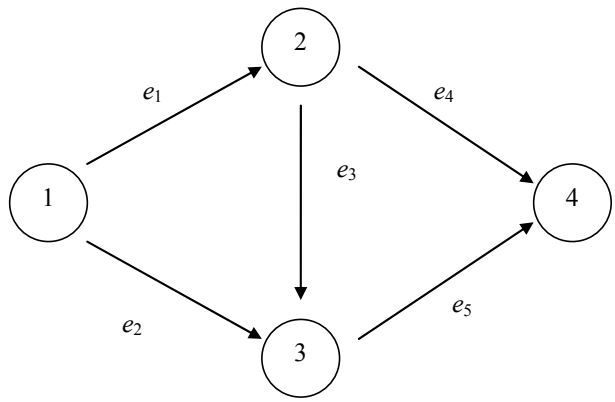

Figure 2. Sample network

In the articles Gaspars (2006a) and Gaspars (2006b) one of us came to the conclusion that the main (but not the only one!) factor affecting the accuracy of the KDA is the way in which variants for compressing the completion time are found. In the original version of the KDA the best set of activities to shorten is chosen from a list of sets containing exactly one activity on each critical path. This approach exposes us to overlooking the cheapest way of accelerating the project by one time unit. In Gaspars (2006a) and Gaspars (2006b) it has been emphasized that the easiest, but not sufficient, way of improving the results, i.e. reducing the compression costs, is to find the cheapest solution among sets containing *at least* (but not exactly) one activity on each critical path. This modification is so natural and obvious that in the next section we will analyze the accuracy of this modified version of the KDA. That means that in order to increase the exactitude of the algorithm it is desirable to consider a more general concept of a cut, as Ford and Fulkerson do (see e.g. Cormen et al., 2001; Ford, Fulkerson, 1962; Gedymin, 1974; Korzan, 1978). To be more specific, they only use the second point of the last definition. This slight modification allows us to use the FFEK[1] algorithm (see e.g. Edmonds, Karp, 1972) for finding the minimal cut instead of an exhaustive search over all the cuts, which makes the algorithm much faster. To

[1] Edmonds – Karp algorithm, one of the adaptations of the Ford – Fulkerson algorithm.

illustrate the difference between these two definitions of the cuts see Figure 2 below. In the case of the original KDA the cuts allowed are $\{e_1, e_2\}$, $\{e_2, e_3, e_4\}$ and $\{e_4, e_5\}$, while in the case of the Ford – Fulkerson algorithm there is one more cut allowed, namely $\{e_1, e_5\}$.

Finally, the version of the Kaufmann – Desbazeille algorithm tested in the next section can be defined as follows:

1. (Initialization) Set the total shortening cost $TC := 0$ and go to step 2.
2. (CPM) Perform the CPM analysis of the project network. If $t_n^I \leq T^d$ then STOP. The current solution is the optimal one[2]. Otherwise go to step 3.
3. (Time reduction / FFEK) Define the following maximum flow problem. Considered arcs in the network are all the critical arcs in the project network. Moreover the capacity of each arc e_i belonging to this network is equal to the first element of the sequence $C^{(i)}$ (if this sequence has at least one element and ∞ otherwise). Using the FFEK method one finds the minimal cut. If the value CV of the minimal cut MC is equal to ∞, then STOP. The deadline problem cannot be solved[3]. Otherwise, go to step 4.
4. (Update) Set $TC := TC + CV$. For each arc $e_i \in MC$ set $t_i := t_i - 1$ and remove the first element of $C^{(i)}$. Go back to step 2.

Note that each time step 4 is performed, the total completion time of the overall project decreases by one. Thus the version of the KDA defined above always stops after a finite number of steps: it either finds the optimal solution or reports the problem to be inconsistent.

6. GOAL OF COMPUTATIONAL EXPERIMENTS

Kaufmann and Desbazeille were only interested in integer realizations of the project time. Therefore, the empirical research will just be related to the deadline problem (see problem 1-2) as the discrete solutions obtained simultaneously constitute solutions for different values of the budgetary parameter in the budget problem. Solutions generated for one of the chosen problem (deadline or budget) suffice to establish the whole time-cost project curve DC (see Figure 1).

In our article we calculate the mean deviation between KD solutions and optimal results. The measure of accuracy (A) is a relative difference between the total project compression costs obtained by the KDA, i.e. C^{KDA}, and the minimum project compression costs C^{min} (the average is calculated over P, i.e. all the test problems, see Equation 5):

---

[2] It is optimal in the sense of the KDA, while it does not have to be the global minimum of the defined deadline problem.

[3] It cannot be solved using the KDA method. It can be easily proved then that the deadline problem is contradictory (inconsistent).

$$A = \frac{1}{P}\sum_{p=1}^{P}\left(\frac{C_p^{KDA} - C_p^{\min}}{C_p^{\min}}\right). \tag{5}$$

The values C_p^{min} constitute the optimal solutions of the following deadline problem:

$$\sum_{i=1}^{m}\sum_{k=1}^{K(i)} c_{ik} y_{ik} \to \min, \tag{6}$$

$$-x_{p(i)} + x_{q(i)} + \sum_{k=1}^{K(i)} y_{ik} - z_i = t_i^n, i = 1,...,m, \tag{7}$$

$$x_1 = 0, x_j \geq 0, j = 1,...,n, x_n \leq T^d, \tag{8}$$

$$0 \leq y_{ik} \leq 1, i = 1,...,m, k = 1,...,K(i), \tag{9}$$

$$z_i \geq 0, i = 1,...,m. \tag{10}$$

with parameters:
m – number of activities (arcs),
n – number of events (nodes),
t_i^n, t_i^c – the normal and crash time of activity e_i, $1 \leq i \leq m$,
$K(i)$ – maximum number of units that activity e_i can be shortened by, $K(i) = t_i^n - t_i^c$,
c_{ik} – cost of shortening activity e_i by k^{th} unit, $c_{ik} \leq c_{i,k+1}$, $1 \leq k < K(i)$,
$p(i)$, $q(i)$ – indices of the starting and end node of arc e_i, $1 \leq p(i) < q(i) \leq n$,
T^d – the desired project completion time,

and variables:
x_j – time of event v_j, $1 \leq j \leq n$,
y_{ik} – binary variable equal to 1 when activity e_i is shortened by k^{th} unit and 0 otherwise,
z_i – final time reserve for activity e_i, $1 \leq i \leq m$.
Each optimal value of the variable x_j always belongs to the interval $[t_j^I, t_j^{II}]$.

In connection with the assumptions made by Kaufmann and Desbazeille (the time parameters are integer) and the fact that we only analyze the deadline problem, the variables in the model (6)-(10) are always integer although there are no additional constraints requiring integer solutions (see Schrijver, 2003). In our experiments we intend to check the accuracy of the KDA for problems similar to the worst case

presented and proved in Anholcer, Gaspars-Wieloch (2011) and for totally random problems.

7. DESCRIPTION OF THE TEST PROBLEMS

Let us recall the details of the case for which the accuracy of the KDA is the worst (see Anholcer, Gaspars-Wieloch, 2011):

1) the network has the structure given in Figure 3 (i.e. the set of arcs consists of all the pairs of nodes of the form (v_j, v_{j+2}), $j = 1, \ldots, n\text{-}2$, and all the pairs of nodes of the form (v_j, v_{j+1}), $j = 1, \ldots, n\text{-}1$) and the number of nodes (n) is even,
2) the normal completion times of activities $<k,k+1>$ equal $t^n_{k,k+1}$ and the normal completion times of activities $<k,k+2>$ equal $2 \cdot t^n_{k,k+1} - 1$ (see Figure 3),
3) the unit shortening costs of activities $<2k, 2k + 1>$, $<2k - 1, 2k>$ and $<k, k + 2>$ are respectively equal to a, b and c where $b > a$ and $c > b\left(\frac{n}{2} - 1\right)$ (see Figure 3),
4) the desired project completion time T^d is equal to $T^n - \left(\frac{n}{2} - 1\right) - 1$, where T^n is the normal project time (i.e. $T^n = (n-1) \cdot t^n_{k,k+1}$)) and $\left(\frac{n}{2} - 1\right)$ is the number of all activities $<2k, 2k + 1>$. This means that $T^d = T^n - \frac{n}{2} = (n-1) \cdot t^n_{k,k+1} - \frac{n}{2}$.

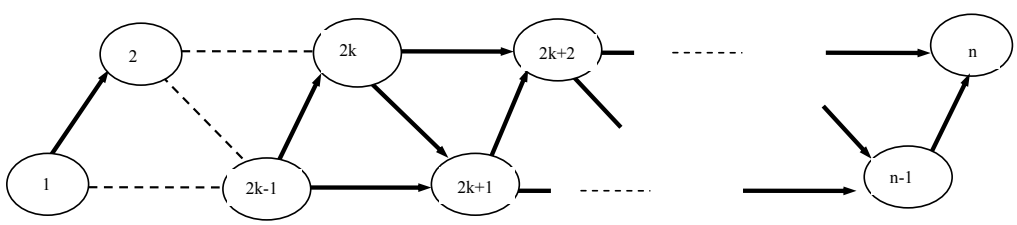

Figure 3. Network type: deterministic

According to the proof given in Anholcer, Gaspars-Wieloch (2011), in such a situation the difference between the total shortening cost given by KDA and the optimal one amounts to $C^{KDA} - C^{\min} = a\left(\frac{n}{2} - 1\right)$. This implies that the gap increases arbitrarily if either n, a or both increase. In particular we have

$$\lim_{n \to \infty}\left(C^{KDA} - C^{\min}\right) = \infty .$$

We use three types of test problems in our experiments. The structure of the first type network is deterministic and is connected to the worst case. Two others are random.

1. Network type 1 (deterministic): The set of nodes consists of n nodes, where n is a parameter. The set of arcs (activities) consists of all the couples of nodes of the form (v_j, v_{j+2}), $j = 1, \ldots, n\text{-}2$, and all the couples of nodes of the form (v_j, v_{j+1}), $j = 1, \ldots, n\text{-}1$ (see Figure 3).
2. Network type 2 (globally random): The set of nodes consists of n nodes, where n is a parameter. There is also one more parameter – D, being the maximum out-degree[4]. The process of generating the arcs consists of two steps. In the first step, for each node v_j, $j = 1, \ldots, n\text{-}D$, we choose uniformly at random D successors from the set $\{v_{j+1}, \ldots, v_n\}$. Each remaining node, i.e. node v_j, $j = n\text{-}D+1, \ldots, n\text{-}1$, is linked with all the nodes from the set $\{v_{j+1}, \ldots, v_n\}$. One may observe that such a way of the network generation does guarantee neither the connectivity of the network nor the uniqueness of the starting event. Thus in the second step all the nodes from the set $\{v_2, \ldots, v_n\}$ are examined. If some of them, say v_j, does not have any predecessor, then an additional arc is added in the following way. A vertex, say v_k, is chosen uniformly at random from the set $\{v_1, \ldots, v_{j-1}\}$, and the new arc is formed as (v_k, v_j).
3. Network type 3 (locally random): The way of generating the arcs is almost the same as in the previous case. The only difference is that in both steps the probabilities of choosing the neighbors are not equal, but the smaller is the difference between the nodes indices, the higher is the probability of joining them with an arc. Namely, the probability of creating an arc joining the pair of nodes (v_j, v_{j+k}) is k times lower than the probability of the creation of the arc between the nodes v_j and v_{j+1}.

In the deterministic case the integer times are chosen uniformly at random from the given range, where the minimal and maximal values are parameters. The ranges are defined separately for the arcs of the type (v_j, v_{j+1}) and the arcs of the type (v_j, v_{j+2}). In the case of random networks there is one basic range defined for all the arcs of the type (v_j, v_{j+1}). For the arcs joining the vertices (v_j, v_{j+k}), $k \geq 1$, the length of the range is calculated as the floor[5] from the product of the basic time and one of the expressions: k (linear growth), $\ln k + 1$ (logarithmic growth), $k^{1/2}$ (square root growth) or $k^{3/2}$ (power growth).

One more time parameter is defined in case of each problem. It is the maximum time reduction, i.e. the maximal number of time units by which the time of each activity may be reduced. This parameter is defined either in units or in percents (in the latter case the resulting times are rounded to integer values if necessary).

---

[4] In fact, in the case of some nodes the maximum out-degree may be finally equal to $D + 1$.

[5] The floor function $\lfloor x \rfloor$ is defined as the greatest integer lower or equal to x: $\lfloor x \rfloor = \max\{y \in \mathrm{Z} \mid y \leq x\}$. E.g. $\lfloor 2.7 \rfloor = 2$, $\lfloor -2.7 \rfloor = -3$, $\lfloor 2 \rfloor = 2$, $\lfloor -2 \rfloor = -2$.

Having defined it, one may define all the possible durations for each activity. For each duration the reduction cost is generated in the following way. The starting reduction cost is a real number chosen uniformly at random from a given range. Then for each duration of the activity the increase of the cost (in comparison to the cost for the previous considered duration) is a real number chosen from another range. As both ends of the ranges have to be positive, the shortening cost sequences are always non-decreasing. In the case of the deterministic network the pair of ranges under consideration is defined separately for each type of arcs: (v_{2j+1}, v_{2j+2}), (v_{2j}, v_{2j+1}) and (v_j, v_{j+2}). In the case of the random ones, there is one range defined for all the arcs.

Now let us analyze in details the generated problems.

In the case of deterministic type network, we have tested only the problems on 10 nodes. We considered four types of test problems, distinguished with the time and basic costs generation schemes. Times on the edges $(k, k + 1)$ were equal to 5 in two first cases (Tables 2 and 3) and chosen randomly from the range [1; 5] in the remaining two (Tables 4 and 5); times on the edges $(k, k + 2)$ equal to 9 and 5 or chosen from [5; 10] and [1; 5], respectively. The basic costs for the arcs $(2k, 2k + 1)$, $(2k - 1, 2k)$ and $(k, k + 2)$ have been chosen respectively from the ranges [1; 2], [2.1; 5], [25; 50] (first and second type) and [1; 5], [1; 5], [1; 5] (third and fourth type), see Figure 3. In all four cases we considered the problems with constant and increasing costs. All the increments were chosen from some intervals as in the random case (the ranges are given in respective tables). In each case two shortening modes were tested.

In the case of the random and locally random networks we have tested the problems on 10 and 50 vertices. The problems have been tested for all the types of the ranges for activities duration (constructed using the linear, logarithmic, square root and power growth function). The maximum out-degree was set to 2.

The basic times (to be multiplied by respective growth functions) have been chosen randomly from the range [0; 5].

In the case of the problems on 50 nodes, the maximum activity compression were set to 2 units or 80%, i.e. the duration of each activity was allowed to be shortened by at most 2 units or at most down to 80% (see Tables 8, 9). Test problems on 10 nodes were examined also for the maximum shortening down to 60% (see Tables 6, 7).

The basic shortening costs (c_0) for each activity have been defined as chosen randomly from the range [1; 5]. For each possible unit of the shortening k the cost has been defined recursively as $c_k = c_{k-1} + d$, where the number d has been chosen every time uniformly at random from the range [0; 5].

For each combination of settings (four growth functions times two or three shortening modes) we tested each problem by calculating the Kaufmann and Desbazeille solution with comparison to the exact one. The computations have been performed for each possible integer shortening time T^d (dozens for each test problem). For each combination of settings 100 problems have been generated and tested.

Notice that problems calculated in this article, even though they always have discrete solutions, are not NP-hard.

8. EMPIRICAL RESULTS

The mean accuracy related to the deterministic problems is presented in Tables 2-5.

Table 2.

Network type: deterministic

Constant unit shortening cost (cost increment = 0).		Variant unit shortening cost. Cost increment $(2k, 2k + 1)$: [0; 2]. Cost increment $(2k - 1, 2k)$: [0; 5]. Cost increment $(k, k + 2)$: [0; 50].	
Short. 2 units	Short. 70%	Short. 2 units	Short. 70%
5.91329	5.16709	4.16203	3.90385

Nodes: 10. Time $(k, k + 1)$: 5. Time $(k, k + 2)$: 9. Basic cost $(2k, 2k + 1)$: [1; 2]. Basic cost $(2k - 1, 2k)$: [2.1; 5]. Basic cost $(k, k + 2)$: [25; 50].

Table 3.

Network type: deterministic

Constant unit shortening cost (cost increment = 0).		Variant unit shortening cost. Cost increment $(2k, 2k + 1)$: [0; 2]. Cost increment $(2k - 1, 2k)$: [0; 5]. Cost increment $(k, k + 2)$: [0; 50].	
Short. 2 units	Short. 70%	Short. 2 units	Short. 70%
0.00000	0.00000	0.00000	0.00000

Nodes: 10. Time $(k, k + 1)$: 5. Time $(k, k + 2)$: 5. Basic cost $(2k, 2k + 1)$: [1; 2]. Basic cost $(2k - 1, 2k)$: [2.1; 5]. Basic cost $(k, k + 2)$: [25; 50].

Table 4.

Network type: deterministic

Constant unit shortening cost (cost increment = 0).		Variant unit shortening cost. Cost increment $(2k, 2k + 1)$: [1; 2]. Cost increment $(2k - 1, 2k)$: [1; 2]. Cost increment $(k, k + 2)$: [1; 2].	
Short. 2 units	Short. 70%	Short. 2 units	Short. 70%
0.57990	0.03260	0.14376	0.05813

Nodes: 10. Time $(k, k + 1)$: [1; 5]. Time $(k, k + 2)$: [5; 10]. Basic cost $(2k, 2k + 1)$: [1; 5]. Basic cost $(2k - 1, 2k)$: [1; 5]. Basic cost $(k, k + 2)$: [1; 5].

Table 5.

Network type: deterministic

Constant unit shortening cost (cost increment = 0).		Variant unit shortening cost. Cost increment ($2k$, $2k + 1$): [1; 2]. Cost increment ($2k - 1$, $2k$): [1; 2]. Cost increment (k, $k + 2$): [1; 2].	
Short. 2 units	Short. 70%	Short. 2 units	Short. 70%
0.84420	0.00000	0.46760	0.00000

Nodes: 10. Time (k, $k + 1$): [1; 5]. Time (k, $k + 2$): [1; 5]. Basic cost ($2k$, $2k + 1$): [1; 5]. Basic cost ($2k - 1$, $2k$): [1; 5]. Basic cost (k, $k + 2$): [1; 5].

This time the accuracy is worse than in the case of randomly generated networks. It is mostly visible in the Table 2, where the numbers reach the level of about 5%. It means that the KDA is not as much useful in the cases where many critical and subcritical paths with a lot of common arcs may appear (see Figure 3). Observe that we defined the deterministic type of network in a very special way in order to make such a situation very likely. However, the poor accuracy of the KDA occurs not only because of the fact that networks are characterized by an extremely peculiar structure and by a huge number of critical and subcritical paths with a lot of common edges. The third characteristic of the networks analyzed in Table 2 is the quite unusual level of shortening costs for each type of activities (compare Table 2 with Tables 4-5). Let us recall this specific case. The unit shortening costs of activities $<2k, 2k + 1>$, $<2k - 1, 2k>$ and $<k, k + 2>$ are respectively equal to a, b and c where $b > a$ and $c > b\left(\frac{n}{2} - 1\right)$.

The mean accuracy for random and locally random problems is given in Tables 6-9.

Table 6.

Network type: locally random

Short. Mode	Linear growth	Power growth	Square root growth	Logarithmic growth
2 units	1.09354	0.54207	0.61875	0.68812
80%	0.00000	0.00000	0.00000	0.00000
60%	0.00052	0.00151	0.03298	0.03353

Nodes: 10. Out-degree: 2. Time: [1; 5]. Basic cost: [1; 5]. Cost increment: [0; 5].

Table 7.

Network type: globally random

Short. Mode	Linear growth	Power growth	Square root growth	Logarithmic growth
2 units	0.24885	0.97609	1.16946	0.45341
80%	0.00000	0.00000	0.00000	0.00000
60%	0.06152	0.00397	0.05415	0.00714

Nodes: 10. Out-degree: 2. Time: [1; 5]. Basic cost: [1; 5]. Cost increment: [0; 5].

Table 8.

Network type: locally random

Short. Mode	Linear growth	Power growth	Square root growth	Logarithmic growth
2 units	0.28675	0.57902	0.34920	0.31711
80%	0.03424	0.00126	0.00510	0.02037

Nodes: 50. Out-degree: 2. Time: [1; 5]. Basic cost: [1; 5]. Cost increment: [0; 5].

Table 9.

Network type: globally random

Short. Mode	Linear growth	Power growth	Square root growth	Logarithmic growth
2 units	0.02317	0.00000	0.55802	0.19056
80%	0.00000	0.00000	0.00000	0.00000

Nodes: 50. Out-degree: 2. Time: [1; 5]. Basic cost: [1; 5]. Cost increment: [0; 5].

As one can see, the accuracy is usually better in networks consisting of more nodes (compare Table 8 with Table 6 and Table 9 with Table 7). The differences between the KD solutions and the exact ones are not very significant – only few of them exceeds the level of 1%, while many of them are less than 10^{-5}%. This proves our supposition that the modified version of KDA behaves quite good in the situation where the network does not contain too much critical paths (which is very likely in our case, as the network is quite sparse).

We tested two kinds of random networks. The ones called *globally random* have arcs distributed in a sense uniformly, while the *locally random networks* have rather a structure close to the D^{th} power of path (i.e. the graph where the vertex v_i is adjacent with the vertices v_j, for which the inequality $|i - j| \leq D$ holds). We hoped that this difference in the structure of the network would lead to some substantial differences in the accuracy of the examined algorithm. The influence of the network structure on the algorithm is not obvious.

In the case of logarithmic growth, the algorithm performs better for globally random networks. In the case of square root growth, it acts better when networks are locally random (with one exception, however in this case – 50 nodes, 80% shortening – the difference is hardly visible). In the case of power growth the algorithm performs better on small locally random and bigger globally random networks. In the case of linear growth, again the algorithm is better for globally random networks (the only exception seems to be not significant). As we can see, in most cases the more ordered structure of the network (locally random) implies a worse performance of the KDA. However, this does not mean that the structure itself has the influence on the way the KDA works. It has to be analyzed together with other features of the problem, in particular the distribution of the shortening costs.

9. CONCLUSIONS

The goal of our article was to check the accuracy of the Kaufmann and Desbazeille algorithm for the worst case described in Anholcer, Gaspars-Wieloch (2011) and for randomly generated problems. As one can notice the difference between the exactitude of the procedure for the deterministic and random networks is quite significant. The KDA is essentially inefficient only in specific cases. Nevertheless we should remember that the computational experiments focused on the modified version of the KDA. The accuracy of the original method is certainly worse.

Poznan University of Economics

REFERENCES

[1] Anholcer M., Gaspars-Wieloch H., (2011), The Efficiency Analysis of the Kaufmann and Desbazeille Algorithm for the Deadline Problem, *Operations Research and Decisions* 2011/2, Wrocław.
[2] Bladowski S., (1970), *Metody sieciowe w planowaniu i organizacji pracy*, PWE, Warszawa.
[3] Bozarth C., Handfield R.B., (2005), *Introduction to Operations and Supply Chain Management*, Prentice Hall, Pearson.
[4] Cormen T. H., Leiserson C. E., Rivest R. L., (2001), *Introduction to Algorithms*, MIT Press, Cambridge.
[5] De P., Dunne E. J., Gosh J. B., Wells C. E., (1995), The Discrete Time-Cost Tradeoff Problem Revisited, *European Journal of Operations Research*, 81, 225–238.
[6] De P., Dunne E. J., Gosh J. B., Wells C. E., (1997), Complexity of the Discrete Time-Cost Trade-Off Problem for Project Networks, *Operations Research*, 45, 302–306.
[7] Edmonds J., Karp R. M., (1972), Theoretical Improvements in the Algorithmic Efficiency for Network Flow Problems, *Journal of the ACM*, 19, 248–264.
[8] Ford L. R., Fulkerson D. R, (1962), *Flows in Networks*, Princeton University Press, Princeton, N. J.
[9] Fusek A., Nowak K., Podleski H., (1967), *Analiza drogi krytycznej* (*CPM i PERT*). *Instrukcja programowana*, PWE, Warszawa.

[10] Gaspars H., (2006), Analiza czasowo-kosztowa (CPM-COST). Algorytm a model optymalizacyjny, *Badania Operacyjne i Decyzje*, 2006, nr 1, Wydawnictwo Politechniki Wrocławskiej, 5–19.
[11] Gaspars H., (2006), Propozycja nowego algorytmu w analizie czasowo-kosztowej przedsięwzięć, *Badania Operacyjne i Decyzje*, 2006, nr 3–4, Wydawnictwo Politechniki Wrocławskiej, 5–27.
[12] Gaspars-Wieloch H., (2009), *Metody optymalizacji czasowo-kosztowej przedsięwzięcia*, (doctoral thesis), Uniwersytet Ekonomiczny w Poznaniu, Poznań.
[13] Gaspars-Wieloch H., (2008), Przegląd wybranych metod skracania czasu realizacji przedsięwzięcia, w: Kopańska-Bródka *D*., (red.)*, Metody i zastosowania badań operacyjnych*, Wydawnictwo Akademii Ekonomicznej w Katowicach.
[14] Gedymin O., (1974), *Metody optymalizacji w planowaniu sieciowym*, PWN, Warszawa.
[15] Giard V., (1991), *Gestion de Projets*, Economica, Paris.
[16] Hendrickson C., Au T., (1989), *Project Management for Construction. Fundamental Concepts for Owners, Engineers, Architects and Builders*, Prentice Hall.
[17] Idźkiewicz A. Z., (1967), *PERT. Metody analizy sieciowej*, PWN, Warszawa.
[18] Kaufmann A., Desbazeille G., Ventura E., (1964), *La Méthode du Chemin Critique*, Dunod, Paris.
[19] Kopańska-Bródka D., (1998), *Wprowadzenie do badań operacyjnych,* wydanie drugie poprawione, Wydawnictwo Akademii Ekonomicznej w Katowicach.
[20] Korzan B., (1978), *Elementy teorii grafów i sieci. Metody i zastosowania*, Wydawnictwa Naukowo-Techniczne, Warszawa.
[21] Kukuła K. (red.), Jędrzejczyk Z., Skrzypek J., Walkosz A., (1996), *Badania operacyjne w przykładach i zadaniach*, PWN, Warszawa.
[22] Muller Y., (1965), *Exercices d'Organisation et de la Recherche Opérationnelle*, Eyrolles, Paris.
[23] Muller Y., (1964), *Organisation et Recherche Opérationnelle*, Paris, Eyrolles.
[24] Panagiotakopoulos D., (1977), A CPM Time-Cost Computational Algorithm for Arbitrary Activity Cost Functions, *INFOR*, 15 (2), 183–195.
[25] Schrijver A., (2003), *Combinatorial Optimization: Polyhedra and Efficiency*, Springer – Verlag Berlin Heidelberg.
[26] Siudak M., (1998), *Badania operacyjne*, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa.
[27] Skutella M., (1998), Approximation Algorithms for the Discrete Time-Cost Tradeoff Problem*, Mathematics of Operations Research*, 23 (4), 909–928.
[28] Trocki M. (red.)*,* Grucza B., Ogonek K., (2003), *Zarządzanie projektami*, PWE, Warszawa.
[29] Waters D., (1998), *A Practical Introduction to Management Science*, Second edition, Addison-Wesley Longman.

DOKŁADNOŚĆ ALGORYTMU KAUFMANNA I DESBAZEILLE W PROBLEMACH OPTYMALIZACJI CZASOWO-KOSZTOWEJ PROJEKTU

Streszczenie

Analiza czasowo–kosztowa jest bardzo ważnym elementem zarządzania projektem. Algorytm Kaufmanna–Desbazeille dla tego problemu jest przez wielu autorów określany mianem dokładnego, lecz w kilku pracach wykazano, iż w niektórych przypadkach stosowanie tej procedury prowadzi jedynie do rozwiązań bliskich optimum. W artykule wyznaczamy średnią dokładność algorytmu dla pewnej liczby sieci o z góry ustalonej bądź losowo wygenerowanej strukturze. Dokładność procedury Kaufmanna i Desbazeille jest najniższa, gdy:

– sieć jest generowana w sposób deterministyczny (parzysta liczba węzłów, sieć składa się z samych łuków łączących sąsiednie węzły, sąsiednie węzły parzyste i sąsiednie węzły nieparzyste, a więc posiada wiele ścieżek krytycznych i podkrytycznych ze wspólnymi łukami),
– każdy typ czynności w tak skonstruowanej sieci ma bardzo specyficzne charakterystyki czasowo-kosztowe.

Struktura sieci ma wpływ na wydajność algorytmu. Powinna być jednak analizowana łącznie z rozkładem jednostkowych kosztów skrócenia czynności.

Słowa kluczowe: analiza czasowo-kosztowa projektów, sieć, ścieżka krytyczna, dokładność algorytmu, skracanie czasu realizacji projektu, krzywe czasowo-kosztowe, minimalizacja kosztu przy zadanym czasie dyrektywnym

ACCURACY OF THE KAUFMANN AND DESBAZEILLE ALGORITHM FOR TIME-COST TRADE-OFF PROJECT PROBLEMS

Abstract

The time-cost tradeoff analysis is a very important issue in the project management. The Kaufmann-Desbazeille method is considered by numerous authors as an exact algorithm to solve that problem, but in some articles it has been proved that for specific network cases the procedure only leads to quasi-optimal solutions. In this paper we calculate the average accuracy of the algorithm for several deterministic and randomly generated networks. The accuracy of the KDA is the worst when:

– the network is generated in a deterministic way (an even number of nodes, the network contains only arcs connecting neighbouring nodes, neighbouring even nodes and neighbouring odd nodes, thus it has many critical and subcritical paths with a lot of common arcs),
– each type of activities in such a network has very specific time-cost characteristics.

The structure of the network has the influence on the performance of KDA. It should be however analyzed together with the distribution of the shortening costs.

Keywords: time-cost tradeoff project analysis (TCTP-analysis), network, critical path, accuracy of the algorithm, project compression time, time-cost curves, deadline problem