B A D A N I A O P E R A C Y J N E I D E C Y Z J E

Nr 3–4 2006

Helena GASPARS*

PROPOZYCJA NOWEGO ALGORYTMU W ANALIZIE CZASOWO-KOSZTOWEJ PRZEDSIĘWZIĘĆ

Autorka pracy nawiązuje do swojego poprzedniego opracowania, w którym pokazano, że algorytm CPM-COST nie zawsze prowadzi do rozwiązania optymalnego. Celem niniejszej pracy jest zaproponowanie nowej metody, która między innymi tym różni się od tradycyjnego podejścia, iż zakłada się konieczność rozpatrywania, w procesie skracania czasu realizacji przedsięwzięcia, wszystkich ścieżek niedopuszczalnych, czyli dróg, których czas trwania przekracza dyrektywny czas realizacji całego przedsięwzięcia. Przedstawioną procedurę porównano z innymi istniejącymi metodami, stosowanymi w analizie czasowo-kosztowej przedsięwzięć.

Słowa kluczowe: *CPM-COST, metoda ścieżki krytycznej, analiza czasowo-kosztowa, czas graniczny, czas dyrektywny, algorytm, ręczne procedury obliczeniowe, podejście dyskretne, małe projekty*

Wstęp

Od kilkudziesięciu lat znana jest metoda ścieżki krytycznej w ujęciu kosztowym, czyli tzw. algorytm CPM-COST. Procedurę tę stosuje się wówczas, gdy decydentowi zależy na przyspieszeniu realizacji projektu opisanego siecią czynności – przy jednoczesnym dążeniu do minimalizacji łącznych kosztów, związanych ze skróceniem czasu trwania wybranych działań. Opis wspomnianego algorytmu można znaleźć m.in. w pracach [3, s. 182–183], [13, s.136]. W późniejszych opracowaniach traktujących o programowaniu sieciowym i zarządzaniu projektem [44, s. 254–255], [27, s. 165–166], autorzy, omawiając kolejne etapy tej metody, powołują się właśnie na pracę [3] i zamieszczają taki sam opis algorytmu (por. także [26, s. 86], [48, s. 430]). W pracy [14] pokazano, iż algorytm CPM-COST jest jedynie procedurą przybliżoną. Celem niniejszej pracy jest przedstawienie koncep-

* Katedra Badań Operacyjnych, Akademia Ekonomiczna w Poznaniu, al. Niepodległości 10, 60-967 Poznań, e-mail: helenagaspars@poczta.onet.pl

cji nowego algorytmu, który został opracowany m.in. na podstawie wniosków, zawartych w artykule [14].

1. Niedoskonałości algorytmu CPM-COST

Procedura algorytmu CPM-COST składa się z następujących etapów (zob. [3], [27], [44]):

1. Zestawić czynności krytyczne, podać gradienty kosztów[1] oraz czasy graniczne.
2. Wyeliminować te czynności krytyczne, których nie można skrócić[2].
3. Skracanie rozpocząć od czynności krytycznej o najniższym gradiencie kosztów.
4. Przy skracaniu czasu trwania czynności starać się skrócić jej czas o jak największą liczbę jednostek czasu, przy czym mogą wystąpić dwa ograniczenia:

– osiągnięcie czasu granicznego dla danej czynności,

– pojawienie się nowej ścieżki krytycznej/ścieżek krytycznych.

5. Jeżeli istnieją dwie (lub więcej) ścieżki krytyczne w sieci, należy skracać czas o tę samą wielkość na wszystkich ścieżkach krytycznych.

Z opisu metody wynika, iż:

- W celu skrócenia czasu realizacji całego przedsięwzięcia należy rozpatrywać jedynie ścieżki krytyczne, ponieważ to one charakteryzują się najdłuższym czasem trwania, a zatem decydują o momencie zakończenia inwestycji.
- Ścieżka, która jest krytyczna bądź stała się nią na skutek kompresji sieci zachowuje w kolejnych iteracjach swoje zerowe zapasy czasu, czyli nie przestaje być krytyczną.
- Wraz z przyspieszaniem momentu zakończenia przedsięwzięcia liczba ścieżek krytycznych w każdej następnej iteracji jest taka sama bądź wzrasta[3],
- Należy skracać wszystkie ścieżki krytyczne o tę samą wielkość, co oznacza, iż przy ustalaniu możliwych wariantów powinny być rozpatrywane tylko niektóre przekroje. Przekrojem sieci krytycznej nazywamy każdy minimalny zbiór łuków, przez które przechodzą wszystkie ścieżki krytyczne. Z opisu algorytmu należy wywnioskować, iż powinniśmy wziąć pod uwagę jedynie kombinacje, stanowiące maksymalne zbiory łuków równoległych. Wariant rozumiany w ten sposób pozwala skrócić wszystkie ścieżki krytyczne o dokładnie jedną jednostkę. Z kolei przekrój jest pojęciem szerszym, gdyż jest nim również każdy racjonalny wariant, zakładający skróce-

[1] Przyjmuje się, że funkcje czas–koszt dla poszczególnych czynności są deterministyczne i niezależne.

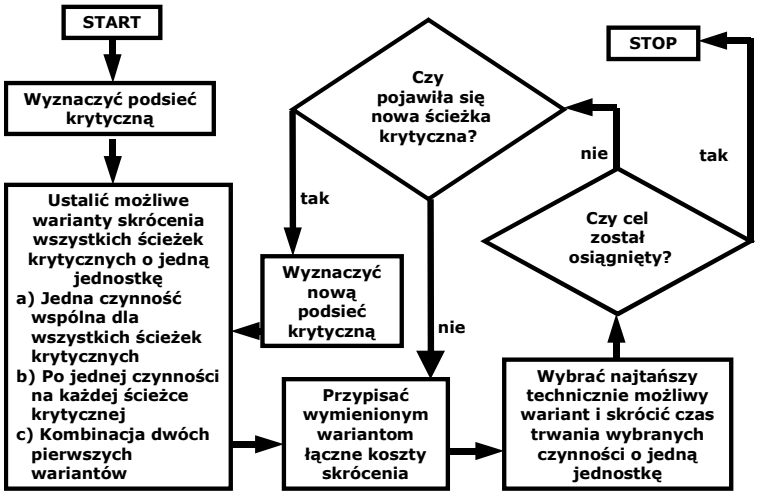
[2] Czasu trwania czynności nie można skrócić, gdy normalny czas wykonania działania jest równy czasowi granicznemu.

[3] Autorzy prac [15, s. 50], [23, s. 82], [29, s. 99], [33, s. 109] oraz [48, s. 427] duży nacisk kładą również na możliwość pojawienia się nowych dróg krytycznych w kolejnych iteracjach.

nie czasu najdłużej trwających ścieżek o przynajmniej jedną jednostkę. Każdy wariant skracający wszystkie najdłuższe drogi o dokładnie jedną jednostkę zawiera się zatem w zbiorze przekrojów sieci krytycznej. Kombinacje, o których mowa w opisie algorytmu CPM-COST, nazwiemy w pracy przekrojami „dokładnymi”[4].

• Chcąc przyspieszyć realizację inwestycji, należy wybrać najtańszy przekrój dokładny[5].

Algorytm CPM-COST można zaprezentować w postaci schematu, przedstawionego na rysunku 1. Skracanie czasu trwania najdłuższych ścieżek sukcesywnie o jedną jednostkę jest konieczne zwłaszcza wówczas, gdy w sieci występuje kilka ścieżek krytycznych. Jeżeli natomiast sieć składa się tylko z jednej drogi krytycznej, której czas trwania jest na dodatek znacznie dłuższy aniżeli czasy wyznaczone dla pozostałych ścieżek, to wybraną(e) czynność(i) krytyczną(e) można w danej iteracji przyspieszyć od razu o więcej niż jedną jednostkę, pamiętając o ograniczeniach, wynikających z ustalonego czasu granicznego oraz dostępnych środków, przeznaczonych na akcelerację działań.



Rys. 1. Algorytm CPM-COST

Sposób ustalania wariantów skracających czas trwania wszystkich ścieżek krytycznych o jedną jednostkę przedstawiono m.in. w pracach [19, s. 303], [33,

[4] Każdy przekrój dokładny jest przekrojem, ale nie każdy przekrój jest przekrojem dokładnym. Dodajmy, że przekrojami są tylko zbiory złożone z czynności rzeczywistych, a nie pozornych, gdyż czas trwania tych ostatnich nie może być skrócony!

[5] Wymienionymi zasadami kierują się również nauczyciele akademiccy, prowadzący zajęcia z programowania sieciowego, czy też zarządzania projektem, o czym świadczą zamieszczane na stronach internetowych wykładowców dydaktyczne opracowania z tego zakresu.

s. 115–117][6]. Osiągnięcie celu wiąże się z uzyskaniem rozwiązania optymalnego jednego z poniższych zadań [43]:

a) Decydent minimalizuje czas realizacji przedsięwzięcia (T), mając na względzie dostępne środki (K*), które może przeznaczyć na akcelerację (*Budget Problem*):

$$T \to \min , \tag{1}$$

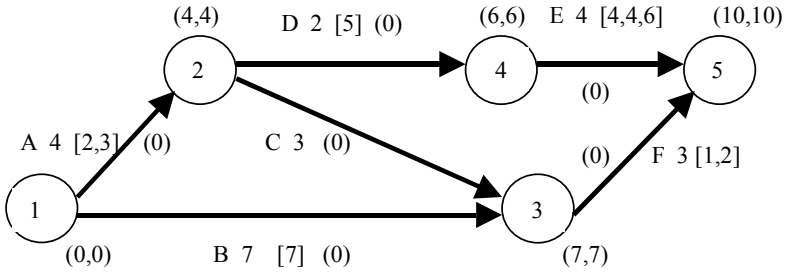
$$K \le K^* . \tag{2}$$

b) Decydent dąży do realizacji inwestycji w czasie nie dłuższym niż zadany czas dyrektywny (T*), przy czym zamierza to uczynić jak najtaniej (*Deadline Problem*) :

$$K \to \min , \tag{3}$$

$$T \le T^* . \tag{4}$$

W pracy [14] przedstawiono jednak sytuacje, w których rozwiązanie – uzyskane za pomocą zaprezentowanego algorytmu – było mniej zadowalające aniżeli wynik, otrzymany na podstawie modelu optymalizacyjnego. W pierwszym przykładzie (rys. 2), zawartym w artykule [14], celem było skrócenie czasu realizacji przedsięwzięcia, składającego się z sześciu czynności (A–F), z 10 do 8 dni. Koszty skrócenia danej czynności o każdy kolejny dzień podano w nawiasach kwadratowych (w tys. zł), a liczba wartości w nich zawartych stanowiła jednocześnie liczbę dni, o którą maksymalnie można było skrócić czas trwania rozpatrywanej czynności. Brak nawiasu kwadratowego przy danym działaniu oznaczał, iż jego skrócenie jest technicznie niemożliwe. W nawiasach okrągłych przy węzłach podano najwcześniejsze możliwe i najpóźniejsze dopuszczalne momenty zajścia zdarzeń, a całkowite zapasy czasu czynności przedstawiono w nawiasach okrągłych obok łuków.



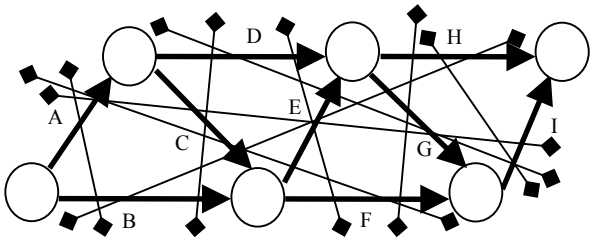
Rys. 2. Wyjściowa sieć w przykładzie 1 ($K \to \min$, $T^* = 8$)

[6] Z trzecim sposobem wyznaczania kombinacji, wymienionym na rys. 1, mamy do czynienia na przykład wówczas, gdy sieć krytyczna składa się z trzech ścieżek: I, II, III i jako wariant przyjmujemy skrócenie jednej czynności wspólnej dla ścieżek I i III oraz jeszcze jednej czynności, która znajduje się tylko na ścieżce II.

Stosując opisaną procedurę generowania rozwiązania optymalnego, otrzymano następujące wyniki. W pierwszej iteracji należało skrócić czas trwania czynności E i F, w kolejnym etapie – również. Łączne koszty skrócenia czasu realizacji przedsięwzięcia wyniosły 11 tys. zł. Tymczasem rozwiązując zadanie na podstawie odpowiednio sformułowanego modelu optymalizacyjnego (zob. [14]), można było dojść do wniosku, iż wariant A+F jest bardziej opłacalny. Dwukrotne skrócenie tych czynności także pozwoliło osiągnąć czas dyrektywny, a całkowite koszty skrócenia wyniosły jedynie 8 tys. zł. Przykład ten wyraźnie pokazuje, iż przy wyborze najtańszego wariantu nie należy ograniczać się do zbiorów czynności, których skrócenie spowoduje jednakowe przyspieszenie wszystkich ścieżek krytycznych. Konieczne jest rozpatrzenie wszystkich możliwych przekrojów.

Twierdzenie 1. W celu skrócenia czasu realizacji przedsięwzięcia z T do $T - \Delta$ możliwie jak najtaniej, należy wziąć pod uwagę nie tylko przekroje dokładne, lecz wszystkie przekroje.

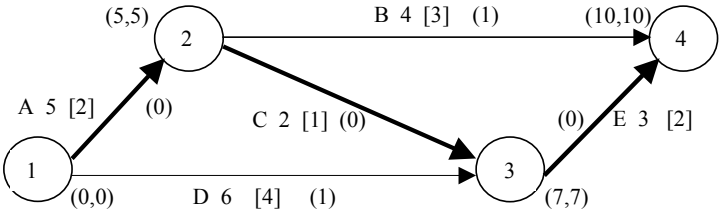
Dowód. Niech B^k oznacza zbiór przekrojów dokładnych, ustalonych na podstawie ścieżek krytycznych sieci, których czas trwania wynosi T, i niech C^k będzie zbiorem wszystkich możliwych przekrojów, wyznaczonych dla tego zestawu ścieżek. Dane są przekroje: $p_1 \in B^k$ oraz $p_2 \in C^k \setminus B^k$ wraz z odpowiadającymi im kosztami skrócenia $K(p_1)$ i $K(p_2)$, przy czym $K(p_1) > K(p_2)$. Niezależnie od wybranego przekroju, skrócenie czasu trwania wszystkich czynności należących do niego o Δ jednostek spowoduje przyspieszenie całej inwestycji o Δ. Ze względu na równanie (3) korzystniej jest jednak wybrać przekrój p_2.



Rys. 3. Możliwe warianty

Przykład pierwszy świadczy o tym, iż raz ustalona ścieżka krytyczna w następnej iteracji niekoniecznie zachowa zerowy zapas czasu. Jeżeli najtańszą kombinacją okaże się zestaw czynności, skracający długość ścieżek krytycznych o przynajmniej jedną jednostkę, to niektóre najdłuższe drogi zostaną skrócone np. dwukrotnie, czyli przestaną być krytyczne! Liczba ścieżek krytycznych może się więc zmniejszyć w kolejnej iteracji, co przeczy założeniom omówionego algorytmu. Warto zauważyć, iż – przy odpowiedniej strukturze sieci – najtańszym przekrojem może się okazać taki zbiór, który zawiera aż $n/2$ bądź $(n + 1)/2$ czynności, wchodzących

w skład ścieżki krytycznej złożonej z n łuków. Stąd w ramach jednej iteracji całkowity zapas czasu danej ścieżki może wzrosnąć nawet o więcej niż dwie jednostki! Na rysunku 3 przedstawiono sieć, składającą się z samych ścieżek krytycznych o czasie T. Jeżeli koszt przekroju {A,E,I} będzie najniższy, ścieżka A-C-E-G-I zostanie skrócona trzykrotnie, mimo iż całe przedsięwzięcie opisane taką siecią zakończy się dopiero w momencie $T - 1$.



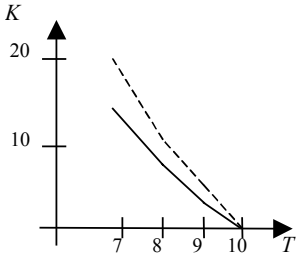
Rys. 4. Wyjściowa sieć w przykładzie 2 ($K \to \min$, $T^* = 8$)

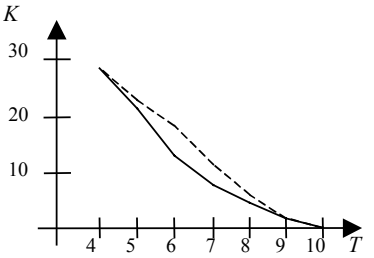
Drugi przykład (rys. 4), opisany w pracy [14], jest znacznie bardziej niepokojący, gdyż dzięki niemu pokazano, iż opieranie się na przekrojach dokładnych, a nie na wszystkich możliwych przekrojach, nie jest jedynym mankamentem algorytmu CPM-COST. Celem zadania było zrealizowanie przedsięwzięcia w ciągu 8 dni. Przyjęto, iż koszt skrócenia o każdy kolejny dzień jest stały dla danej czynności. Czasy graniczne wynosiły odpowiednio: 2, 1, 1, 3 i 1 dzień. Początkowo jedyną ścieżką krytyczną była droga A-C-E.

Mimo rozszerzenia listy możliwych wariantów o kombinacje zakładające skrócenie ścieżek krytycznych o przynajmniej jedną jednostkę, otrzymany wynik sugerował skrócenie w pierwszej kolejności czasu trwania czynności C, a następnie – czynności A i E, podczas gdy rozwiązanie uzyskane za pomocą modelu wskazało jedynie czynności A i E. Skrócenie czynności C okazało się zatem zbyteczne, a nieco zmodyfikowana wersja algorytmu znów wygenerowała droższe rozwiązanie (5 tys. zł) aniżeli model optymalizacyjny (4 tys. zł). Dodajmy, że bazując tylko na przekrojach dokładnych, wyznaczylibyśmy strategię, której koszt wynosi aż 6 tys. zł (I iteracja: C, II iteracja: B+E), przy czym żadna ścieżka nie trwałaby krócej niż 8 dni, czyli z długości dróg wynikałoby, iż nie ma czynności, która zostałaby skrócona niepotrzebnie. Nadmiarowe przyspieszenie danej czynności może być więc możliwe tylko wówczas, gdy w poprzednich iteracjach wykorzystano przekrój niebędący przekrojem dokładnym.

Twierdzenie 2. Jeżeli w trakcie skracania czasu realizacji przedsięwzięcia do momentu T^* najtańszym wariantem w przynajmniej jednej iteracji okaże się przekrój, zbudowany na podstawie ścieżek krytycznych i nie będący przekrojem dokładnym, to istnieje ryzyko, że ostatecznie niektóre czynności zostaną skrócone niepotrzebnie.

Dowód. Niech $T^* = T^1 - 2$, gdzie T^1 oznacza początkowy czas trwania ścieżek krytycznych. Załóżmy, że w I etapie ($T^1 \rightarrow T^1 - 1$) najtańszym wariantem jest przekrój p_1 taki, że $p_1 \in B_1^k$, gdzie B_1^k stanowi zbiór przekrojów dokładnych ustalonych dla ścieżek, które są krytyczne na początku I iteracji. Ponadto, niech S_1^k i $S_2^k(p_1)$ będą zbiorami ścieżek krytycznych na początku I i II iteracji[7]. Ponieważ $p_1 \in B_1^k$, jest więc oczywiste, iż $S_1^k \subseteq S_2^k(p_1)$. Przyjmijmy, że $p_1 = \{a_1, a_2, ..., a_h, ..., a_n\}$, gdzie a_1, a_2, …, a_n oznaczają czynności krytyczne, wchodzące w skład przekroju p_1, przy czym $a_h \in s^1$ i $a_h \notin S_1^k \setminus \{s^1\}$, a s^1 jest ścieżką krytyczną, która spełnia następujący warunek: $s^1 \notin S_2^k(p_1) \setminus S_1^k$. Załóżmy również, że najtańszą kombinacją w II etapie ($T^2 \rightarrow T^2 - 1$) jest przekrój p_2 taki, że $p_2 \in C_2^k(p_1) \setminus B_2^k$, gdzie $C_2^k(p_1)$ jest zbiorem wszystkich przekrojów, wyznaczonych dla zbioru $S_2^k(p_1)$. Niech p_2' będzie przynajmniej dwuelementowym podzbiorem zbioru p_2 takim, że $p_2' \in s^1$. Jeżeli zmianę czasu realizacji przedsięwzięcia oznaczymy jako ΔT, a wyrażenie $\Delta t(S_1^k)$ będzie określać łączną zmianę czasu trwania dróg będących ścieżkami krytycznymi na początku pierwszej iteracji, to po II etapie $\Delta t(S_1^k) \geq \Delta T$, a $\Delta t(s^1) > \Delta T$. Skoro skrócona czynność a_h należy tylko do ścieżki s^1, to jej ponowne wydłużenie nie naruszy warunku $\Delta t(S_1^k) \geq \Delta T$, który gwarantuje uzyskanie czasu T^*, a więc przyspieszenie działania a_h nie jest konieczne.





Rys. 5. Krzywe czasowo-kosztowe dla przykładu 1 **Rys. 6.** Krzywe czasowo-kosztowe dla przykładu 2

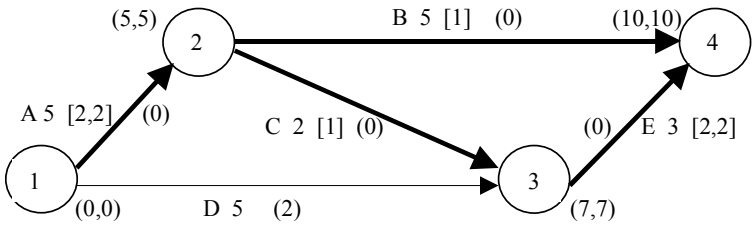
W pracy [14] zwrócono również uwagę na to, iż zaproponowane rozwiązania zastosowania algorytmu i modelu optymalizacyjnego dla przykładu drugiego będą identyczne, gdy $T^* = 9$ dni. W obu przypadkach skrócenie czasu trwania czynności C o 1 dzień okaże się najlepszą strategią. Zauważmy więc, że – w celu uzyskania

[7] Zbiór ścieżek krytycznych w kolejnych iteracjach zależy od wybranego przekroju w poprzednim etapie. Dlatego dla II iteracji przyjęto zapis $S^k_2(p_1)$.

coraz krótszych czasów realizacji projektu – algorytm dołącza do zbioru skróconych już czynności kolejne działania, model dobiera natomiast optymalny zestaw czynności, stosownie do ustalonego czasu dyrektywnego. Innymi słowy, skumulowana kombinacja dla czasu T niekoniecznie musi być podzbiorem skumulowanej kombinacji ustalonej dla czasu $T - 1$. Na rysunkach 5 i 6 przedstawiono krzywe czasowo-kosztowe dla obu przykładów, skonstruowane na podstawie wyników uzyskanych za pomocą pierwotnej wersji algorytmu (linia przerywana) i modelu (linia ciągła). Długość krzywych jest zdeterminowana czasami granicznymi poszczególnych czynności.

W pracy [14] wysunięto przypuszczenie, że stosowanie algorytmu sprawi, iż każdej kolejnej iteracji może towarzyszyć coraz większe odchylenie między wynikiem otrzymanym a optymalnym. Przykład 1 świadczy o możliwości wystąpienia takiego zagrożenia. Im czas realizacji przedsięwzięcia jest krótszy, tym większa jest różnica kosztów. W przykładzie 2 z kolei czasy graniczne zostały ustalone na bardzo niskim poziomie, co sprawia, iż do pewnego poziomu parametru T^* algorytm rzeczywiście generuje coraz gorsze rozwiązania aniżeli model optymalizacyjny. Przy dalszym skracaniu możliwe jest natomiast wyrównanie wyników. Można by zatem stwierdzić, że skoro docelowo obie metody prowadzą do tych samych rezultatów, to wytykanie niedoskonałości algorytmu CPM-COST jest pozbawione sensu. Wiadomo jednak, iż nie zawsze decydent może sobie pozwolić na maksymalne przyspieszenie inwestycji ze względu na ograniczone środki. Dlatego pośrednie realizacje są równie istotne. Warto także zaznaczyć, iż krzywe czasowo-kosztowe, dotyczące wyników wygenerowanych przez model optymalizacyjny, uzyskano rozwiązując odrębne zadania dla różnych poziomów T^*. Kolejne otrzymane punkty nie ilustrują więc rozwiązań dla poszczególnych iteracji.

Poza niedoskonałościami algorytmu CPM-COST, omówionymi w pracy [14], można jeszcze wymienić jedno, dość istotne zagrożenie. Załóżmy, że czas realizacji przedsięwzięcia opisanego na rysunku 7 należy skrócić o 3 dni. W pierwszej iteracji mamy do wyboru dwie kombinacje, charakteryzujące się najniższym kosztem (A oraz B + C).



Rys. 7. Wyjściowa sieć w przykładzie 3 ($K \rightarrow \min$, $T^* = 7$)

W przypadku poprawnie działających algorytmów nie ma znaczenia, która zmienna (wariant) o najbardziej korzystnym wskaźniku zostanie wybrana(y)[8]. Wybór danej zmiennej (wariantu) może, co najwyżej, wpłynąć na liczbę kroków, lecz nie determinuje ostatecznej wartości funkcji celu, która dla rozwiązania końcowego powinna być zawsze taka sama, niezależnie od etapów pośrednich. Algorytm CPM-COST nie daje jednak takiej gwarancji. W tabeli 1 zebrano uzyskane rezultaty końcowe w zależności od kombinacji wybranej w danej iteracji.

Tabela 1

Iteracje uzyskane za pomocą algorytmu CPM-COST (przykład 3)

T	I opcja		II opcja		III opcja	
	Wariant	Koszt	Wariant	Koszt	Wariant	Koszt
9	A CB BE	2 2 3	A CB BE	2 2 3	A CB BE	2 2 3
8	A CB BE	2 ∞ ∞	A CB BE	2 2 3	A CB BE	2 2 3
7	AD BCD BE	∞ ∞ ∞	AD BCD BE	∞ ∞ 3	AD BCD BE	∞ ∞ ∞
Rezultat	Brak rozwiązania dopuszczalnego		Otrzymano rozwiązanie: łączny koszt = 7		Brak rozwiązania dopuszczalnego	

Mimo iż we wszystkich trzech przypadkach kierowano się minimalizacją kosztów skrócenia, tylko jedna opcja okazała się skuteczna. W pozostałych zbyt wcześnie skrócono czynność B, a ta z kolei tworzy wraz z działaniem E jedyny dopuszczalny wariant w ostatniej iteracji. Po raz kolejny dochodzimy więc do wniosku, iż na problem należy spojrzeć całościowo i że optymalizacja w ramach danej iteracji nie musi wcale prowadzić do optimum globalnego. Sposób postępowania powinien być od samego początku zdeterminowany zadanym czasem dyrektywnym. Warto zwrócić uwagę na jeszcze jedną kwestię. W tabeli 1 zamieszczono jedynie przekroje dokładne. Gdyby przekrój A+E został dołączony w ostatniej iteracji do listy wariantów, uzyskanie rozwiązania dopuszczalnego (choć nie optymalnego!) byłoby możliwe zarówno dla opcji I, jak i opcji III. Rozpatrywanie wszystkich możliwych przekrojów zwiększa zatem szansę uzyskania jakiegokolwiek rozwiązania dopuszczalnego.

Na koniec wyobraźmy sobie, że czynność B w przykładzie 3 można skrócić aż o dwa dni, lecz koszt skrócenia tego działania o drugi dzień wynosi 3 jednostki. W tej sytuacji łączne koszty przyspieszenia inwestycji o 3 dni wyniosą odpowiednio 9, 7 i 9 jednostek. Tym razem uzyskano wprawdzie rozwiązania dopuszczalne, niezależnie od

[8] Por. np. z algorytmem Little'a czy metodą potencjałów w zagadnieniu transportowym.

przyjętego sposobu skracania czynności, lecz tylko jedno z nich stanowi optymalną strategię, mimo iż we wszystkich trzech przypadkach wybierano najtańsze warianty.

Twierdzenie 3. Gdy przy skracaniu czasu realizacji przedsięwzięcia wybierane są przekroje budowlane na podstawie sieci krytycznej, a w danej iteracji istnieją dwa najtańsze warianty, łączny koszt skrócenia do momentu T^* może zależeć od tego, którą najtańszą kombinację wybierzemy.

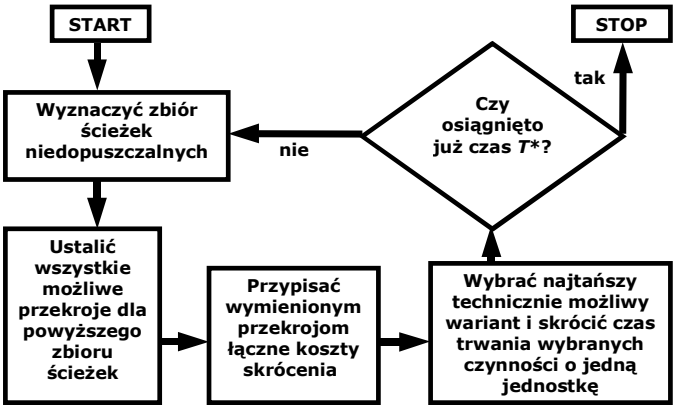
Dowód. Niech $T^* = T^1 - 2$, gdzie T^1 oznacza początkowy czas trwania ścieżek krytycznych sieci. Załóżmy, że w I etapie ($T^1 \rightarrow T^1 - 1$) najtańszymi wariantami są przekroje p_1 i p_2 ustalone dla S_1^k, czyli wszystkich ścieżek, które są krytyczne na początku I iteracji. Przekrojom p_1 i p_2 odpowiadają koszty $K(p_1)$ i $K(p_2)$, gdzie $K(p_1) = K(p_2)$[9]. Niech ponadto $S_2^k(p_1)$ i $S_2^k(p_2)$ będą zbiorami ścieżek krytycznych na początku II etapu w zależności od wybranego przekroju w etapie poprzednim, przy czym $S_2^k(p_1) = S_2^k(p_2)$. Przyjmijmy, że czynność $a_d \in p_1$ i że $a_d \notin p_2$ oraz że k_d^i oznacza koszt skrócenia czasu trwania czynności a_d o i-tą jednostkę. Niech przekrój r_1 będzie jedynym możliwym przekrojem ustalonym dla zbioru S_2^k takim, że $a_d \in r_1$. Wiadomo również, że $k_d^1 < k_d^2$, a koszt skrócenia czasu pozostałych czynności, dla których $t^n > t^g$ (czas normalny przewyższa czas graniczny) jest stały. Koszt odpowiadający wariantowi r_1 jest więc zależny od przekroju wybranego w iteracji poprzedniej: $K_{r1}(p_1) > K_{r1}(p_2)$, a to oznacza, iż koszt przyspieszenia inwestycji do momentu $T^* = T^1 - 2$ również zależy od wyboru dokonanego w I etapie: $K^{T-2}(p_1, r_1) = K(p_1) + K_{r1}(p_1) > K^{T-2}(p_2, r_1) = K(p_2) + K_{r1}(p_2)$.

2. Modyfikacja pierwotnej wersji algorytmu CPM-COST

Stanisław Bladowski w pracy [3], komentując zaprezentowany algorytm stwierdza, iż „opisana metoda grafoanalityczna jest daleko prostsza niż metody matematyczne algorytmu Fulkersona lub programowania liniowego, parametrycznego, a ogólna dokładność optymalizacji programu jest praktycznie taka sama". Mając na uwadze zaobserwowane rozbieżności między rozwiązaniami generowanymi przez algorytm i model optymalizacyjny, należy jednak zaznaczyć, iż owa dokładność jest w dużej mierze zdeterminowana strukturą sieci, kosztami ponoszonymi przy skracaniu czasu trwania poszczególnych czynności oraz ustalonymi czasami granicznymi.

[9] Zakładamy, że żaden z przekrojów nie prowadzi do przyspieszenia inwestycji o więcej niż jedną jednostkę.

Dokonane obserwacje pozwalają przypuszczać, iż niepoprawne działanie przedstawionego algorytmu wynika z wyboru wariantu na podstawie niekompletnej bądź nieodpowiedniej listy możliwych kombinacji. Wnioski zebrane w poprzednim rozdziale posłużą nam do opracowania bardziej efektywnego algorytmu (rys. 8).



Rys. 8. Zmodyfikowana wersja algorytmu ($K \rightarrow \min$, $T \leq T^*$)

Ścieżką niedopuszczalną jest każda droga, której czas trwania $T_s > T^*$. Konstruowanie wariantów na podstawie zarówno ścieżek krytycznych (których czas T jest równy momentowi zakończenia całej inwestycji), jak i dróg podkrytycznych z całkowitym zapasem czasu mniejszym od wyrażenia ($T - T^*$), stanowi jedną z najistotniejszych różnic między omawianą nową koncepcją algorytmu a tradycyjnym podejściem. Rozpatrywanie wszystkich ścieżek niedopuszczalnych, już na samym początku rozwiązywania problemu, jest konieczne, ponieważ to właśnie z powodu pominięcia dróg podkrytycznych wybór wariantów spośród dotychczasowych list kombinacji, ustalonych dla każdej iteracji, nie zawsze prowadził do wyznaczenia najbardziej opłacalnej strategii.

Łączne koszty skrócenia czynności z danego wariantu obliczamy sumując koszty przyspieszenia wszystkich działań, wchodzących w skład tego przekroju. Przy wyborze zestawu czynności, których czas trwania powinien być skrócony o jedną jednostkę, należy kierować się poniższymi zasadami:

a) Gdy najniższe koszty związane są z przekrojem nie w pełni krytycznym, skracamy jedynie czas trwania czynności krytycznych należących do tej kombinacji. Skracanie czynności niekrytycznych może z jednej strony szybciej zmniejszyć zbiór ścieżek niedopuszczalnych, lecz z drugiej strony istnieje ryzyko, iż w ostatecznym rozrachunku przyspieszenie takich działań okaże się zbędne, a przez to droższe aniżeli wynikałoby to z rozwiązania optymalnego. Za skracaniem tylko działań krytycznych przekroju przemawia jeszcze jeden argument. Otóż samo przyspieszenie czynności

krytycznych jest wystarczające, by czas realizacji przedsięwzięcia został skrócony o jednostkę, co wobec warunku (3) tym bardziej skłania do skracania tylko krytycznej części przekroju.

b) Jeżeli dwa przekroje charakteryzują się tym samym najniższym kosztem, przy czym jeden z nich składa się z samych czynności krytycznych, a drugi z jednej przynajmniej czynności niekrytycznej, warto skorzystać z pierwszego wariantu, gdyż ten pozwala szybciej uzyskać rozwiązanie dopuszczalne.

c) Gdy istnieją dwa najtańsze przekroje, z których żaden nie jest w pełni krytyczny, korzystniej jest wybrać ten zestaw, dla którego suma kosztów skrócenia czasu trwania czynności krytycznych ($k^k(p)$) jest niższa.

d) Pomijamy te przekroje, dla których $k^k(p) = \infty$. Wariant, dla którego $k^k(p) \neq \infty$ i $k^{nk}(p) = \infty$ (gdzie $k^{nk}(p)$ oznacza sumę kosztów skrócenia czynności niekrytycznych przekroju), wykorzystujemy tylko wtedy, gdy dla pozostałych kombinacji w danej iteracji $k^k(p) = \infty$.

Stosowanie wymienionych zasad pozwala nam wybrać warianty o najniższym koszcie i o minimalnej liczbie czynności, których czas trwania należy skrócić, by osiągnąć czas T^*.

Twierdzenie 4. Skracanie czasu realizacji przedsięwzięcia na podstawie wszystkich możliwych przekrojów, ustalonych dla całego zbioru ścieżek niedopuszczalnych, zapobiega skracaniu czynności, których przyspieszenie nie jest konieczne do osiągnięcia czasu T^*.

Dowód. Niech $T^* = T^1 - 2$. Załóżmy, że w I etapie ($T^1 \rightarrow T^1 - 1$) najtańszym wariantem jest przekrój p_1 taki, że $p_1 \in C_1^{nd} \setminus B_1^{nd}$, gdzie C_1^{nd} stanowi zbiór przekrojów ustalonych dla zbioru ścieżek niedopuszczalnych na początku I iteracji (S_1^{nd}), a B_1^{nd} jest zbiorem przekrojów dokładnych, wyznaczonych dla tegoż zbioru. Niech K^j oznacza zbiór czynności krytycznych w j-tej iteracji, a NK^j – zbiór czynności niekrytycznych. Przyjmijmy, że $p_1 = \{a_1, a_2, ..., a_h, ..., a_n\}$, gdzie $a_1, a_2, ..., a_h \in K^1$, $a_{h+1}, ..., a_{n-1}, a_n \in NK^1$. Niech p_1' będzie przynajmniej 2-elementowym podzbiorem krytycznym zbioru p_1 takim, że $p_1' \in s^1$, gdzie s^1 jest ścieżką spełniającą warunek: $s^1 \in S_1^k$. Wybór przekroju p_1 oznacza, iż po I iteracji mogą wystąpić trzy sytuacje:

A. $s^1 \notin S_2^k$, $s^1 \notin S_2^{nd}$, $t(s^1) = T^* < T^2 = T^1 - 1$, gdzie T^2 jest czasem realizacji przedsięwzięcia na początku II iteracji.

B. $s^1 \in S_2^k$, $s^1 \notin S_2^{nd}$, $t(s^1) = T^2 = T^1 - 2 = T^*$ (koniec obliczeń).

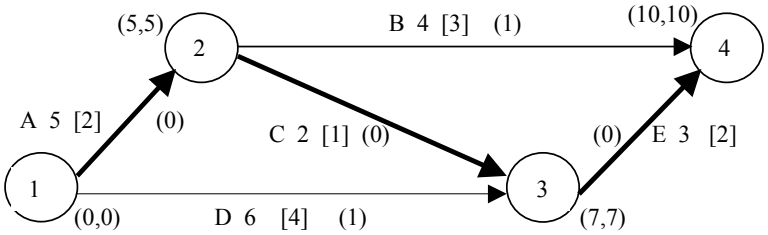
C. $s^1 \notin S_2^k$, $s^1 \notin S_2^{nd}$, $t(s^1) < T^2 = T^1 - 2 = T^*$ (koniec obliczeń).

Jeżeli po I etapie czas $T - 2$ nie zostanie jeszcze osiągnięty (przypadek A), to dowolny przekrój r ustalony dla II etapu będzie taki, że każda jego czynność krytyczna $a_k \in S_2^{nd} \setminus \{s^1\} \vee \; a_k \in S_2^{nd} \cap \{s^1\}$. Skoro przekrój r nie będzie zawierał czynności,

która by należała tylko do ścieżki s^1, możemy mieć pewność, że żadne zbędne działanie nie zostanie skrócone[10].

Z technicznego punktu widzenia zmodyfikowana wersja algorytmu tym różni się od algorytmu CPM-COST, że w miarę zbliżania się do zadanego czasu T^* przekroje są konstruowane dla coraz mniejszej liczby ścieżek, gdyż zbiór dróg niedopuszczalnych maleje w trakcie obliczeń. Stosując tradycyjne podejście, wyznaczamy przekroje dokładne dla ścieżek krytycznych, a liczba takich ścieżek zazwyczaj wzrasta. Skracanie czasu realizacji przedsięwzięcia zgodnie z udoskonaloną procedurą sprowadza się do rozpatrywania iteracji jakby w odwrotnej kolejności aniżeli w przypadku przyspieszania momentu zakończenia inwestycji za pomocą pierwotnej wersji algorytmu CPM-COST.

Twierdzenie 5. Gdy przy skracaniu czasu realizacji przedsięwzięcia wybierane są przekroje budowlane na podstawie ścieżek niedopuszczalnych, wzrastają szanse uzyskania najtańszego rozwiązania dopuszczalnego dla momentu T^* (por. dowód twierdzenia 3).



Rys. 9. Wyjściowa sieć w przykładzie 2 ($K \rightarrow \min$, $T^* = 4$)

W tabeli 2 przedstawiono sposób generowania poszczególnych iteracji przez nową wersję algorytmu dla przykładu 2 (rys. 9) przy założeniu, iż czas dyrektywny wynosi 4 dni[11]. Oczywiście, w przypadku zmodyfikowanej procedury, wybór czynności, których czas trwania należy w danym etapie skrócić, jest ściśle związany z docelowym czasem dyrektywnym. Tymczasem jeśli stosujemy pierwotną wersję algorytmu, to zestaw działań wybranych w rozpatrywanej iteracji jest zawsze taki sam niezależnie od tego, jaki czas realizacji przedsięwzięcia należy ostatecznie uzyskać (tab. 3). Z zestawienia w tabeli 3 wynika, iż nowa wersja algorytmu może wskazać w danym etapie wariant droższy aniżeli kombinacja wyznaczona tradycyjnym podejściem, gdyż zmodyfikowana metoda koncentruje się na efekcie finalnym, podczas gdy założeniem algorytmu CPM-COST jest ustalenie rozwiązania optymalnego dla danej iteracji, które niekoniecznie musi być najlepszą strategią z punktu widzenia wyniku końcowego.

[10] W ogólniejszym przypadku (np. gdy $T^* = T^1 - 3$) możliwe jest, by czynność należąca tylko do ścieżki s^1 weszła w skład przekroju r, lecz w związku z tym, iż $s^1 \notin S_2^k$, byłaby to czynność niekrytyczna, a czas trwania niekrytycznych działań przekroju nie jest skracany.

[11] W zadaniu przyjęto, że czasy graniczne wynoszą odpowiednio 2, 1, 1, 3 i 1 dzień.

Tabela 2

Iteracje uzyskane za pomocą zmodyfikowanego algorytmu (przykład 2, $T^* = 4$ dni)

Czas wyjściowy	Ścieżki niedopuszczalne		Przekroje ustalone na podstawie zbioru ścieżek niedopuszczalnych	Wybrane czynności	Łączny koszt skrócenia	Czas uzyskany
	krytyczne	niekrytyczne				
10	A-C-E	A-B D-E	A+D = 6 A+E = 4 B+C+D = 8 B+E = 5	A, E	4	8
8	A-B A-C-E D-E	-	A+D = 6 A+E = 4 B+C+D = 8 B+E = 5	A, E	4+4 = 8	7
7	A-B D-E	A-C-E	A+D = 6 A+E = ∞ B+C+D = 8 B+E = ∞	A, D	8+6 = 14	6
6	A-B D-E	A-C-E	A+D = ∞ A+E = ∞ B+C+D = 8 B+E = ∞	B, D	14+7 = 21	5
5	A-B A-C-E D-E	–	A+D = ∞ A+E = ∞ B+C+D = 8 B+E = ∞	B, C, D	21+8 = 29	4

Tabela 3

Zestawienie wyników wygenerowanych przez pierwotną i nową wersję algorytmu[12]

Iteracja	Pierwotna wersja algorytmu		Nowa wersja algorytmu		Nowa wersja algorytmu	
	T^*	Łączny koszt	T^*	Łączny koszt	T^*	Łączny koszt
10–9	Dowolny	1	9	1	4	4
9–8	Dowolny	6	8	4	4	4
8–7	Dowolny	11	7	8	4	8
7–6	Dowolny	17	6	14	4	14
6–5	Dowolny	23	5	21	4	21
5–4	Dowolny	29	4	29	4	29

Zauważmy, iż korzystając z nowej metody, pomijamy etap $T = 9$ (tab. 2). Zaletą tej procedury jest możliwość przyspieszenia momentu zakończenia przedsięwzięcia nawet o kilka jednostek w ramach danej iteracji, mimo że czynności z wybranego wariantu skracamy tylko raz. Skorzystajmy ponownie z sieci, opisanej na rysunku 3.

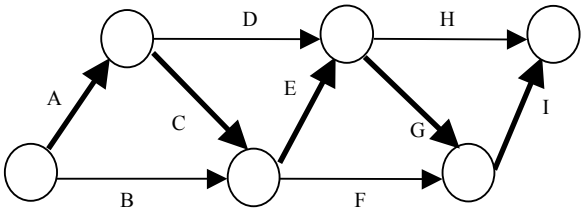
[12] Symbol T^* oznacza czas dyrektywny.

Tym razem jednak przyjmijmy, że tylko ścieżka A-C-E-G-I jest krytyczna (rys. 10). Dane o sieci zawiera tabela 4.

Tabela 4

Dane o sieci z rys. 3

Czynności	A	B	C	D	E	F	G	H	I
Czas trwania (w dniach)	5	9	5	9	5	9	5	9	5
Koszt skrócenia o jeden dzień	1	3	2	2	1	2	1	3	1



Rys. 10. Wyjściowa sieć w przykładzie 4 ($K \rightarrow \min$, $T^* = 22$)

Jeżeli czas dyrektywny wyniesie 22 dni, wszystkie ścieżki będą niedopuszczalne. Dzięki skróceniu czasu trwania czynności z najtańszego przekroju (A+E+I) o jedną jednostkę otrzymamy od razu rozwiązanie dopuszczalne, a zarazem optymalne, mimo iż początkowy czas realizacji przedsięwzięcia był dłuższy od zadanego aż o 3 dni (tab. 5).

Tabela 5

Długość ścieżek dla planu początkowego i optymalnego

Ścieżki	Czas trwania ścieżek	
	przed I iteracją	po I iteracji
A-D-H	23	22
A-D-G-I	24	22
A-C-E-G-I	25	22
A-C-E-H	24	22
A-C-F-I	24	22
B-E-H	23	22
B-E-G-I	24	22
B-F-I	23	22

Możliwość pominięcia niektórych etapów pojawia się również wówczas, gdy dany przekrój można wykorzystać p razy z rzędu. Symbol p^l oznacza liczbę jednostek, o którą warto od razu skrócić czynności a_j^l z przekroju l:

$$p^l = \min\left\{\min_{a_j^l}\{t_j - t_j^g + z_j^c\}, \min_s\left\{E\left(\frac{T_s - T^* + Z_s^l}{w_s^l}\right)+1\right\}\right\}^{13}, \tag{5}$$

gdzie:

t_j – aktualny czas trwania j-tej czynności,

t_j^g – graniczny czas trwania j-tej czynności,

z_j^c – całkowity zapas czasu j-tej czynności,

T_s – czas trwania ścieżki niedopuszczalnej s,

T^* – ustalony czas dyrektywny,

Z_s^l – suma całkowitych zapasów czasu czynności należących jednocześnie do l-tego przekroju i do ścieżki s,

w_s^l – liczba czynności należących jednocześnie do l-tego przekroju i do ścieżki s.

Czynności krytyczne z wybranego wariantu można skrócić p^l razy, natomiast liczba jednostek, o którą należy przyspieszyć j-te działanie niekrytyczne z tego przekroju wynosi $\max\{0, p^l - z_j^c\}$. Przy skracaniu czasu realizacji przedsięwzięcia opisanego w przykładzie 2 przekrój A+E oraz B+C+D są najtańszymi wariantami w dwóch sąsiednich iteracjach. Zastosowanie wzoru (5) może zatem jeszcze bardziej przyspieszyć obliczenia.

Nową metodę można stosować zarówno wówczas, gdy koszty skrócenia czasu trwania danego działania o każdą kolejną jednostkę są takie same, jak i wtedy, gdy owe koszty są różne w zależności od tego, o którą jednostkę czas trwania wybranej czynności jest skracany.

Procedura zaprezentowana w tym rozdziale znajduje zastosowanie w zadaniach o postaci (3)–(4). Gdy celem decydenta jest minimalizacja czasu realizacji przedsięwzięcia przy zadanym koszcie dyrektywnym, liczba jednostek, o którą należy przyspieszyć moment zakończenia inwestycji nie jest z góry znana. Wiadomo jedynie, że skracając czas realizacji projektu o kolejne jednostki, nie można przekroczyć dostępnych środków. W przypadku zadania (1)–(2), nie jesteśmy więc w stanie określić początkowego zbioru ścieżek niedopuszczalnych, na podstawie którego ustalana jest lista przekrojów. Warto wówczas wygenerować rozwiązania dla różnych momentów zakończenia przedsięwzięcia i wybrać ten punkt, leżący na krzywej czasowo-kosztowej, który spełnia warunki (1)–(2).

[13] Wyrażenie $E((T_s - T^* + Z^l_s)/w^l_s)$ oznacza cechę ilorazu $(T_s - T^* + Z^l_s)/w^l_s$. Ze wzoru (5) można korzystać tylko wtedy, gdy koszt skrócenia czasu trwania poszczególnych czynności jest stały.

3. Ocena nowej wersji algorytmu

Zarówno pierwotna, jak i nowa wersja algorytmu może być wykorzystywana przy ustalaniu harmonogramu projektów, charakteryzujących się małą liczbą czynności. Obie metody są ręcznymi procedurami obliczeniowymi, które można stosować niezależnie od tego, czy koszt skrócenia czasu trwania czynności o każdą kolejną jednostkę jest stały czy zmienny, co jest z pewnością ich atutem, gdyż w wielu pracach analizowane są jedynie sytuacje, w których funkcja kosztów spełnia określone założenia[14]. Złożoność obu technik zależy m.in. od budowy sieci, obrazującej realizację przedsięwzięcia oraz od różnicy między rzeczywistym a dyrektywnym momentem zakończenia inwestycji. Techniczna różnica między tymi procedurami polega na tym, iż w algorytmie CPM-COST skracane są czynności leżące na ścieżkach krytycznych, nowa wersja natomiast zakłada akceleracją działań krytycznych niekoniecznie należących do najdłuższych dróg.

Zmodyfikowany algorytm ma tę przewagę nad pierwotną wersją procedury, iż generuje rozwiązania nie gorsze od algorytmu CPM-COST, pozwala uniknąć skrócenia czynności, których przyspieszenie mogłoby w kolejnych iteracjach okazać się zbyteczne[15]. Nowy algorytm nie znajduje natomiast bezpośredniego zastosowania w zadaniach (1)–(2). Ponadto wyników uzyskanych dla pośrednich iteracji, wykonywanych w celu skrócenia czasu realizacji przedsięwzięcia do T^*, nie należy traktować jako rozwiązania optymalne dla momentów $T^* + 1$, $T^* + 2$, …

Warto podkreślić, że poza algorytmem opisanym w pracy [3], można znaleźć w literaturze wiele innych ręcznych procedur obliczeniowych, mających na celu wygenerowanie postaci krzywej czasowo-kosztowej dla małych projektów. Są to jednak metody przybliżone, z których część powstała w sposób intuicyjny[16].

---

[14] Jest liniowa [3], [6], [8], [13], [24], [26], [38], [39], [45], wypukła [28], [33], [42] bądź wklęsła [9].

[15] Zob. twierdzenia i przykłady omówione w poprzednich rozdziałach.

[16] Metodą przybliżoną jest na przykład omówiony w pracy [3] algorytm tablicowy, opracowany na podstawie tych samych założeń, co algorytm CPM-COST. Dlatego też działa on niepoprawnie. Do tej grupy algorytmów można zaliczyć metodę SAM (*Siemens Approximation Method*) [41] i jej udoskonaloną wersję, zaproponowaną przez Goyala [18]. Obie te metody mają zastosowanie tylko wówczas, gdy jednostkowy koszt skrócenia jest stały dla danej czynności. Nadają się one ponadto do bardzo małych projektów ze względu na konieczność iteracyjnego analizowania każdej ścieżki sieci oddzielnie. Metody Siemensa i Goyala są podobne do zmodyfikowanej wersji algorytmu CPM-COST, gdyż w obu tych procedurach wybór czynności, których czas trwania należy skrócić, jest ściśle zdeterminowany docelowym czasem dyrektywnym inwestycji. Kolejnym przybliżonym algorytmem, niewymagającym (choć niewykluczającym) użycia komputera jest metoda Panagiotakopoulosa [36], która przyjmuje bardziej realne postaci funkcji kosztów niż metoda Siemensa czy Goyala. Z praktycznego punktu widzenia jest ona jednak dość złożoną procedurą, gdyż przypomina metodę cechowania węzłów, stosowaną również w pracy [11].

Gdy optymalne czasy trwania czynności nie muszą być całkowite, a każdej czynności odpowiada ciągła liniowa funkcja kosztu zależnego od czasu (np. [45]), problem LTCT[17] rozwiązywany jest w czasie wielomianowym [43]. W przypadku liniowych bądź przedziałami liniowych funkcji kosztu można stosować np. podejście Fulkersona, łączące metodę maksymalnego przepływu w sieci z programem dualnym [11], [12], bądź programowanie parametryczne Kelley'a [25]. Założenie o wypukłej i ciągłej postaci funkcji kosztu czyni jednak te metody mało użyteczne (por. [36])[18]. Ze względu na to, iż dyskretna wersja tego zagadnienia (znacznie bardziej powszechna w praktyce [1], [20], [21], [30], [33], [35], [36]), znana jest jako problem *NP*-trudny, rozwiązywany w czasie wykładniczym [7][19], podejmowane są próby opracowania przybliżonych metod generowania rozwiązania dla problemu dyskretnego w czasie wielomianowym. Wymagają one jednak zastosowania komputera [43, s. 915][20]. Poza tym niektóre z nich znajdują zastosowanie tylko wówczas, gdy koszty skracania czasów trwania działań o kolejne jednostki są stałe. Dlatego opracowanie dostatecznie sprawnego algorytmu dla wersji dyskretnej wydaje się nadal sprawą otwartą, o czym wspominają także Czerwiński i Ignasiak w pracy [6].

Algorytmem najbardziej zbliżonym do procedury zaproponowanej w niniejszej pracy jest metoda minimalnego cięcia, autorstwa Phillipsa i Dessouky'ego (LTCT-Solver) [34], [38]. Stanowi ona udoskonaloną wersję algorytmów Kelley'a i Fulkersona[21]. Minimalne cięcie to każdy minimalny zbiór łuków, dzielący podsieć krytyczną na dwa rozłączne zbiory zdarzeń S i U takie, że $s \in S$, $u \in U$ i $S \cup U = W$, gdzie

[17] *Linear Time–Cost Tradeoff Problem.*

[18] Zakładanie, iż funkcja kosztu dla poszczególnych czynności powinna być wypukła, jest kwestionowane przez wielu autorów. Ich zdaniem wystarczy, by punkty leżące na krzywej czasowo-kosztowej dla działań stanowiły rozwiązania Pareto-optymalne, które z kolei nie wykluczają wklęsłej postaci rozpatrywanej funkcji.

[19] Jeżeli przyjmiemy, że w skład diagramu sieciowego wchodzi n czynności (j = 1, .., n), a każdej z nich przypisano m_j opcji czasowych, to istnieje $\prod_{j=1}^{n} m_j$ możliwych rozwiązań. Tylko wtedy, gdy sieć składa się z niezależnych ścieżek, problem dyskretny można rozwiązać w czasie wielomianowym.

[20] Ten algorytm polega na ustaleniu optimum dla zadania programowania liniowego, a następnie na znalezieniu najbliższego rozwiązania dopuszczalnego zadania programowania dyskretnego. Wadą owej procedury jest możliwość przyjęcia maksymalnie dwóch opcji czasowych dla każdej czynności. Do grupy metod przybliżonych działających w czasie wielomianowym można zaliczyć także algorytmy dokonujące podziału dużego projektu na subprojekty, dla których wyznaczane są strategie optymalne (zob. [43, s. 922–925] oraz [33, s.130]).

[21] Ponieważ zaproponowane na początku lat sześćdziesiątych ubiegłego stulecia metody Fulkersona i Kelley'a wymagały odpowiedniego przygotowania matematycznego, bardzo szybko podjęto próbę opracowania, na podstawie tychże metod, algorytmów, z których mogliby korzystać np. inżynierowie. Zanim Phillips i Dessouky przedstawili swoją koncepcję, Prager przygotował procedurę, która miała być zrozumiała nie tylko dla przedstawicieli środowiska naukowego [39]. Wykorzystał w tym celu prezentację graficzną, przypominającą wykres Gantta. Metoda okazała się jednak dość złożona i czasochłonna.

s jest źródłem sieci, *u* – jej ujściem, a *W* stanowi zbiór wszystkich zdarzeń grafu[22]. Z definicji wynika, że minimalne cięcie to zbiór tych wszystkich łuków, które zarówno wchodzą (*forward arcs*), jak i wychodzą (*backward*, *reverse arcs*) ze zbioru *U*. Z kolei przekrój składa się z samych czynności typu *forward*. Phillips i Dessouky proponują, by – po wybraniu najtańszego cięcia – czasy trwania czynności *forward* zostały skrócone, a czynności *backward* (jeżeli trwają krócej niż czas normalny) – wydłużone o jednostkę. Choć idea algorytmu jest prosta (por. [22]), a sama procedura zwraca zawsze optimum i to nie tylko dla czasu docelowego, lecz też wszystkich pośrednich etapów, to jednak pod kilkoma względami ustępuje ona udoskonalonej wersji algorytmu CPM-COST[23].

Oprócz całej gamy algorytmów, stanowiących w większości przypadków metody heurystyczne tworzone z myślą o małych projektach, drugą kategorią technik stosowanych w analizie czasowo-kosztowej są metody programowania liniowego [45], całkowitoliczbowego [20], [30], [32], [35], dynamicznego [5], [7], [21], [40][24] i nieliniowego [28]. W przypadku niektórych z nich nie przyjmuje się nawet żadnych założeń co do postaci funkcji kosztu dla czynności [32], [35]. Zastosowanie metod pro-

[22] W pracy [33] cięcie zdefiniowane jest jako zbiór łuków taki, że każda ścieżka krytyczna ma przynajmniej jeden łuk należący do tegoż cięcia. Znalezienie minimalnego cięcia odbywa się w czasie $O(nm\log(n^2/m))$, gdzie *n* oznacza liczbę zdarzeń, a *m* liczbę czynności podsieci krytycznej [16]. Sposób usprawniający szukanie minimalnego cięcia przedstawiono w artykule [46].

[23] Po pierwsze, możliwości skrócenia czasu realizacji przedsięwzięcia o kilka jednostek w ramach jednej iteracji są bardzo ograniczone. Po drugie, metoda minimalnego cięcia została opracowana przy założeniu, że koszt skrócenia czasów trwania poszczególnych działań jest stały. Po trzecie, metoda ta dopuszcza sytuację, w której czas trwania czynności skróconej w *j*-tej iteracji będzie wydłużony w *k*-tej iteracji, gdzie $k > j$. Wyobraźmy sobie, że kierownik projektu, w momencie gdy już przedsięwziął odpowiednie kroki, by skrócić czas realizacji przedsięwzięcia do momentu T^*, podejmuje decyzję o dodatkowym przyspieszeniu inwestycji do czasu $T^* - t$. W tej sytuacji może się okazać, iż kierownik niepotrzebnie przeznaczył środki na skrócenie czynności, która zgodnie z ustaloną strategią optymalną dla czasu $T^* - t$ nie powinna być wcale przyspieszona. Wydłużanie w trakcie obliczeń czynności skróconych wcześniej należy traktować jako pewnego rodzaju korektę, która z praktycznego punktu widzenia nie jest jednak wygodna. O potrzebie korekcyjnego wydłużania działań skróconych w poprzednich iteracjach mowa jest też w pracach: [3, s. 204], [15, s. 50, 53], [18], [33], [41], [43]. Jednak w niektórych przypadkach nadmierne skrócenie czynności we wcześniejszych etapach nie jest wynikiem wyboru przekroju nie będącego przekrojem dokładnym, lecz jest konsekwencją skrócenia czynności o więcej niż jedną jednostkę w ramach danej iteracji, gdy jednostkowe przyspieszenie czasu jej trwania nie było możliwe [33] lub wydawało się mało opłacalne [41].

[24] Metodę Butchera [5] można użyć tylko dla sieci o bardzo prostej budowie. Z kolei algorytm Robinsona [40] wymaga każdorazowo innego sposobu postępowania w zależności od struktury sieci. Zaletą programowania dynamicznego jest to, że wartości nieliniowej funkcji kosztów nie wymagają aproksymacji, gdyż wystarczy je odpowiednio stablicować. Zdaniem Robinsona, programowanie dynamiczne posiada zalety zarówno metod heurystycznych, jak i algorytmów dokładnych. Metody heurystyczne pozwalają przyjąć bardziej realne funkcje kosztów, lecz są tylko metodami przybliżonymi, algorytmy dokładne natomiast generują strategie optymalne, ale ograniczają się do bardzo prostych zależności między czasem trwania czynności a kosztem jego skrócenia.

gramowania matematycznego prowadzi do optimum, lecz rozwiązywanie tego typu zadań, nawet z pomocą komputera, jest niezwykle czasochłonne ze względu na konieczność wprowadzenia wielu zmiennych i warunków, a to oznacza, że tak naprawdę nadają się one do planowania małych projektów. Algorytmy heurystyczne z kolei mogą okazać się szybsze, lecz niestety mniej dokładne [6], [10], [41].

Zauważmy, iż punktem wyjścia dla wszystkich dotychczas wymienionych metod jest założenie, że akceleracja przedsięwzięcia może się odbywać tylko przez skracanie czasu trwania wybranych czynności. Autorzy prac [4], [17, s. 170], [31, s. 137], [37, s. 217], [49, s. 185] proponują zupełnie inny sposób skracania czasu realizacji inwestycji, polegający m.in. na zmniejszeniu zakresu zadań wykonywanych w ramach danej czynności, na dzieleniu czynności na krótsze działania równolegle, bądź na jednoczesnym wykonywaniu tych czynności, które według pierwotnego planu miały być realizowane sekwencyjnie. Taka metoda też wiąże się z ponoszeniem dodatkowych kosztów, a ponadto może wpłynąć na obniżenie jakości wykonania projektu. Przyjęcie założenia o niepodzielności czynności [43] i o braku możliwości dokonywania jakichkolwiek zmian w strukturze projektu oznacza, iż wspomnianą metodę kompresji sieci należy w naszych rozważaniach pominąć.

Długą listę metod wykorzystywanych w analizie czasowo-kosztowej zamykają specjalne programy komputerowe, mające zastosowanie w dziedzinie efektywnego zarządzania projektem (np. *Microsoft Project*, *Lindo*, *GAMS*, *Optimum*, *Primavera*) [2], [30]. Są one przeznaczone do planowania i nadzorowania realizacji projektu.

Ponieważ celem niniejszej pracy było opracowanie ręcznej procedury obliczeniowej, oceny efektywności zaproponowanego algorytmu dokonano więc głównie na podstawie analizy pozostałych technik, które nie wymagają użycia komputera. Wykazano, iż zaproponowana metoda generuje rozwiązania nie gorsze od wyników uzyskiwanych za pomocą algorytmu CPM-COST. W związku z tym, że zmodyfikowana wersja oparta jest na założeniach podobnych to tych, którymi kierowali się Phillips i Dessouky, twórcy procedury zawsze zwracającej optimum, mamy prawo przypuszczać, iż dokładność przedstawionej procedury jest zbliżona do dokładności metody minimalnego cięcia. Nowa wersja algorytmu CPM-COST jest pozbawiona wielu wad nie tylko innych metod heurystycznych, ale również niektórych metod programowania matematycznego. Przede wszystkim można ją stosować dla dowolnej deterministycznej struktury sieci, a koszty jednostkowe skrócenia czasu trwania danej czynności mogą leżeć zarówno na liniowej, jak i wypukłej czy wklęsłej krzywej czasowo-kosztowej. Konstrukcja algorytmu pozwala ponadto znacznie ograniczyć liczbę iteracji, potrzebnych do uzyskania rozwiązania, a zasady opracowane dla tejże metody są dość proste, co sprawia, iż ma ona także walor dydaktyczny.

Bibliografia

[1] ABRAMOV S.A., MARINICZEW M.L., POLJAKOW P.D., *Setiewyje metody planirowanija i uprawlenija*, Sowietskoje Radio, Moskwa 1966.

[2] ABDELSALAM H.M., BAO H.P., *Solving the Project Time-Cost Trade-Off Problem through an Integrated Engineering-Computation Environment*, Proceedings of the Eleventh Annual Conference of the Production and Operations Management Society, POM-2000, April 1–4, San Antonio, Texas 2000.

[3] BLADOWSKI S., *Metody sieciowe w planowaniu i organizacji pracy*, PWE, Warszawa 1970.

[4] BRANDENBURG H., *Zarządzanie projektami*, Wydawnictwo Akademii Ekonomicznej w Katowicach, 2002.

[5] BUTCHER W.S., *Dynamic program for project cost-time curves*, Proceedings ASCE, 1967, 93(CO1).

[6] CZERWIŃSKI Z., *Moje zmagania z ekonomią*, Wydawnictwo Akademii Ekonomicznej w Poznaniu, 2002.

[7] DE P., DUNNE E.J., GHOSH J.B., WELLS C.E., *Complexity of the discrete time-cost trade-off problem for project networks*, Operations Research, 1997, 45, s. 302–306.

[8] ELMAGRAPHY S.E., SALEM A., *Optimal linear approximation in project compression*, IIE Transactions, 1984, 16(4), s. 339–347.

[9] FALK J.E., HOROWITZ J.L., *Critical path problems with concave cost-time curves*, Management Science, 1972, 19, s. 446–455.

[10] FONDHAL J.W., *A Non-Computer Approach to the Critical Path Method for the Construction Industry*, Stanford University, Stanford 1961.

[11] FORD L.R., FULKERSON D.R, *Przepływy w sieciach*, tłum. M. Wycech-Łosiowa, PWN, Warszawa 1969.

[12] FULKERSON D.R., *A network flow computation for project cost curves*, Management Science, 1961, 7(2), s. 167–178.

[13] FUSEK A., NOWAK K., PODLESKI H., *Analiza drogi krytycznej (CPM i PERT). Instrukcja programowana*, PWE, Warszawa 1967.

[14] GASPARS H., *Analiza czasowo-kosztowa (CPM-COST). Algorytm a model optymalizacyjny*, Badania Operacyjne i Decyzje, 2006, nr 1, s. 5–19.

[15] GEDYMIN O., *Metody optymalizacji w planowaniu sieciowym*, PWN, Warszawa 1974.

[16] GOLDBERG A., TARJAN R.E., *A new approach to the maximum flow problem*, J. Assoc. Comput. Mach., 1988, 35, s. 921–940.

[17] GOODMAN L.J., LOVE R.N., *Project Planning and Management. An integrated Approach*, Pergamon Press, New York 1980.

[18] GOYAL S.K., *A note on „A simple CPM time-cost tradeoff algorithm"*, Management Science, 1975, 21(6), s. 718–722.

[19] GUZIK B. (red.), *Ekonometria i badania operacyjne*, MD 51, AE, Poznań 1999.

[20] HARVEY R.T., PATTERSON J.H., *An implicit enumeration algorithm for the time/cost trade-off problem in project network analysis*, Found. Control Engineering, 1979, 4, s. 107–117.

[21] HINDELANG T.J., MUTH J.F., *A dynamic programming algorithm for Decision CPM networks*, Operations Research, 1979, 27(2), s. 225–241.

[22] ICMELI O., ERENGUC S.S., ZAPPE C.J., *Project Scheduling Problems: A Survey*, International Journal of Operations and Production Management, 1993, 13(11), s. 80–91.

[23] IDŹKIEWICZ A.Z., *PERT. Metody analizy sieciowej*, PWN, Warszawa 1967.

[24] IGNASIAK E., *Programowanie sieciowe*, PWE, Warszawa 1972.

[25] KELLEY J.E., *Critical path planning and scheduling: mathematical basis*, Operations Research, 1961, 9(3), s. 296–320.

[26] KOPAŃSKA-BRÓDKA D., *Wprowadzenie do badań operacyjnych*, wyd. II popr., Wydawnictwo Akademii Ekonomicznej w Katowicach, 1998.
[27] KUKUŁA K., JĘDRZEJCZYK Z., SKRZYPEK J., WALKOSZ A., *Badania operacyjne w przykładach i zadaniach*, PWN, Warszawa 1996.
[28] LAMBERSON L.R., HOCKING R.R., *Optimum time compression in project scheduling*, Management Science, 1970, 16(10), s. 597–606.
[29] LEVY F.K., THOMPSON G.L., WIEST J.D., *The ABC's of Critical Path Scheduling*, Harvard Business Review, 1963, 41, s. 98–108.
[30] LIU L., BURNS S.A., FENG C.-W., *Construction time-cost tradeoff analysis using LP/IP hybrid method*, Journal of Construction Engineering and Management, 1995, 121(4), s. 446–453.
[31] LOCK D., *Podstawy zarządzania projektami*, tłum. M. Ciszewska, M. Sosnowski, PWE, Warszawa 2003.
[32] MEYER W.L., SHAFFER L.R., *Extensions of the Critical Path Method Through the Application of Integer Programming*, Department of Civil Engineering, University of Illinois, 1963.
[33] MODER J.J., PHILLIPS C.R., *Project Management with CPM and PERT*, Reinhold Publishing Corporation, New York 1964.
[34] MÖHRING R.H., SCHULZ A.S., STORK F., UETZ M., *On project scheduling with irregular starting time costs*, Operations Research Letters, 2001, 28, s. 149–154.
[35] MOUSSOURAKIS J., HAKSEVER C., *Flexible Model for Time/Cost Tradeoff Problem*, Journal of Construction Engineering and Management, 2004, s. 307–314.
[36] PANAGIOTAKOPOULOS D., *A CPM time-cost computational algorithm for arbitrary activity cost functions*, INFOR, 1977, 15(2), s. 183–195.
[37] PHILLIPS J., *Zarządzanie projektami IT. Poznaj najskuteczniejsze metody zarządzania przedsięwzięciami informatycznymi*, tłum. M. Lipa, P. Pilch, M. Szczepaniak, Helion, Gliwice 2005.
[38] PHILLIPS S.J., DESSOUKY M.I., *Solving the Time/Cost Tradeoff Problem using the Minimum Cut Concept*, Management Science, 1977, 24(4), s. 393–400.
[39] PRAGER W., *A structural method of computing project cost polygons*, Management Science, 1963, 9(3), s. 394–404.
[40] ROBINSON D.R., *A dynamic programming solution to cost-time tradeoff for CPM*, Management Science, 1975, 22(2), s. 158–166.
[41] SIEMENS N., *A simple CPM time-cost tradeoff algorithm*, Management Science, 1971, 17(6), s. 354–363.
[42] SIEMENS N., GOODING C., *Reducing project duration at minimum cost: a time/cost trade-off algorithm*, OMEGA, 1975, 3, s. 569–581.
[43] SKUTELLA M., *Approximation algorithms for the discrete time-cost tradeoff problem*, Mathematics of Operations Research, 1998, 23(4), s. 909–928.
[44] TROCKI M., GRUCZA B., OGONEK K., *Zarządzanie projektami*, PWE, Warszawa 2003.
[45] TRZASKALIK T., *Modelowanie optymalizacyjne*, Absolwent, Łódź 2000.
[46] TUFEKCI S., *A Flow-preserving Algorithm for the Time-Cost Trade-Off Problem*, AIII Transactions, 1982, 2(3), s. 109–113.
[47] VANHOUCKE M., *New computational results for the discrete time/cost trade-off problem with time-switch constraints*, Vlerick Working Papers, 2002, 18, s. 3–33.
[48] WATERS D., *A practical introduction to management science*, II ed., Addison-Wesley Longman, 1998.
[49] WYSOCKI R.K., MCGARY R., *Efektywne zarządzanie projektami*, wyd. III, tłum. T. Rychoń, M. Szolc, Helion, Gliwice 2005.

A conception of a new algorithm for the project time-cost analysis

The author presents again examples of activity-on-arcs networks already given in her previous contribution which revealed that in some cases the algorithm based on the critical path method with a time-cost analysis (CPM-COST) does not provide any optimal solution. The author also presents other defects of this method which have not been previously discussed. That is why, a new hand computational procedure for small projects is proposed. Its main assumption consists in taking into consideration, during the project acceleration, both critical paths as well as subcritical paths, the duration of which is longer than the desired project completion time. Additionally, the author puts emphasis on the fact that the project compression does not necessarily require that each critical path be shortened by exactly the same number of units. Sometimes other solutions are cheaper and just as efficient. The new technique is compared with existing heuristic and mathematical methods.

Keywords: *CPM-COST, critical path method, time-cost analysis, crash time, (target) desired project completion time, algorithm, hand computational procedures, discrete approach, small projects*