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 4 2008

Zbigniew TARAPATA*

ALGORYTMY HARMONOGRAMOWANIA ZSYNCHRONIZOWANEGO PRZEMIESZCZANIA WIELU OBIEKTÓW

W pracy przedstawiono algorytmy wyznaczania harmonogramu zsynchronizowanego przemieszczania wielu obiektów. Zdefiniowano problem harmonogramowania przemieszczania w postaci dwukryterialnego nieliniowego zadania optymalizacji. Omówiono sposób rozwiązania sformułowanego problemu dwukryterialnego poprzez poszukiwanie rozwiązania leksykograficznego. Podano dwa algorytmy harmonogramowania zsynchronizowanego przemieszczania wielu obiektów oraz opisano ich własności. Szczegółowo omówiono konieczne i wystarczające warunki umożliwiające otrzymanie rozwiązania optymalnego. Oszacowano złożoności obliczeniowe prezentowanych algorytmów oraz przedyskutowano ich własności. Zasadę działania algorytmów zilustrowano przykładami.

Słowa kluczowe: *harmonogramowanie i synchronizacja przemieszczania, drogi najkrótsze, drogi rozłączne, wielokryterialne problemy dróg najkrótszych, algorytmy harmonogramowania*

1. Wprowadzenie

Harmonogramowanie zsynchronizowanego przemieszczania wielu obiektów wykorzystywane jest między innymi w symulatorach walki, komputerowych grach symulacyjnych (w podsystemie planowania i symulacji przemieszczania obiektów (jednostek) [5], [8], [9], [13]), ale również przy synchronizacji pracy wielu przemieszczających się agentów (obiektów), jak np. w problemach sterowania ramionami wielu niezależnych robotów [7], czy w koordynacji mobilnych obiektów (np. robotów) współpracujących ze sobą [2]. Szczególnie interesujące może być poszukiwanie takich rozwiązań, które pozwalają osiągnąć pewne cele (jeden lub więcej), takie jak:

* Zakład Badań Operacyjnych i Wspomagania Decyzji, Instytut Systemów Informatycznych, Wydział Cybernetyki, Wojskowa Akademia Techniczna, ul. Kaliskiego 2, 00-908 Warszawa, e-mail: zbigniew.tarapata@wat.edu.pl

osiągnięcie punktów docelowych przez wszystkie obiekty w ściśle określonym czasie, równoczesne przybycie wszystkich obiektów do punktów pośrednich, spełnienie pewnych dodatkowych ograniczeń (np. brak zatrzymywania się w punktach pośrednich) itp. Jednym ze wspomnianych zastosowań omawianego problemu jest planowanie i synchronizacja przemieszczania obiektów w komputerowych grach symulacyjnych [12], [13]. W grach typu człowiek–komputer wykorzystywana jest technika generowania przez komputer zachowania drugiej strony (tzw. systemy CGF (ang. *Computer Generated Forces*) [8], [13] lub SAF (ang. *Semi-Automated Forces*) [6]). Zachowanie odpowiedniego ugrupowania w czasie działań jest bardzo istotne z punktu widzenia realizacji celu działania. Dla przykładu, każdy obiekt podlegający przemieszczaniu (podczas ataku, przegrupowania itp.), będący elementem grupy obiektów (jednostek, zgrupowań, kolumn transportowych), musi „trzymać" ugrupowanie, tzn. musi przemieszczać się biorąc pod uwagę położenie innych obiektów grupy zgodnie z pewnym wzorcem ugrupowania. Dlatego też praca skupia się na algorytmach rozwiązywania problemów harmonogramowania zsynchronizowanego przemieszczania wielu obiektów i jest kontynuacją pracy [14], w której przedstawiono szereg modeli harmonogramowania.

W rozdziale 2 podano podstawowe oznaczenia i definicje stosowane w pracy. W rozdziale 3 sformułowano problem harmonogramowania zsynchronizowanego przemieszczania wielu obiektów. Rozdział 4 zawiera opis algorytmów rozwiązania sformułowanego problemu wraz z opisem ich własności oraz własności rozwiązań otrzymywanych z tych algorytmów. Podano również przykłady ilustrujące własności prezentowanych algorytmów.

2. Definicje i oznaczenia

W celu zaprezentowania algorytmów harmonogramowania zsynchronizowanego przemieszczania wielu obiektów przypomnimy podstawowe definicje i oznaczenia oraz modele harmonogramowania przedstawione w pracy [14]. Zakładamy, że struktura środowiska przemieszczania (sieci dróg lub terenu podzielonego na heksagony, kwadraty itd. [8], [9]) reprezentowana jest przez graf Berge'a $G = \langle V_G, A_G \rangle$, gdzie $V = |V_G|$, V_G – zbiór wierzchołków grafu (skrzyżowań lub kwadratów (ich środków) terenu), A_G – zbiór łuków grafu, $A_G \subset V_G \times V_G$, $A = |A_G|$. Zakładamy, że dla każdego łuku grafu G dysponujemy wartością $d_{n,n'}$ funkcji d, która opisuje odległość terenową między wierzchołkami n oraz n'. Dysponujemy K obiektami, które chcemy przemieścić z wektora $s = (s_1, s_2, ..., s_K)$ wierzchołków początkowych do wektora $t = (t_1, t_2, ..., t_K)$ wierzchołków końcowych w G. W dalszych rozważaniach przyjmiemy następujące oznaczenia:

$$I_k(s_k, t_k) = I_k = (i^0(k) = s_k, i^1(k), ..., i^r(k), ..., i^{R_k}(k) = t_k), \tag{1}$$

$$T_k(I_k) = T_k(\tau^0(k), \tau^1(k), ..., \tau^r(k), ..., \tau^{R_k}(k)), \tag{2}$$

$$V_k(I_k) = V_k(v_{i^0(k), i^1(k)}, v_{i^1(k), i^2(k)}, ..., v_{i^{R_k-1}(k), i^{R_k}(k)}), \tag{3}$$

gdzie:

I_k – wektor wierzchołków opisujących drogę dla k-tego obiektu,

$\underset{m\in\{1,...,R_k\}}{\forall}(i^{m-1}(k), i^{m-1}(k)) \in A_G$,

$i^r(k)$ – r-ty wierzchołek na drodze k-tego obiektu,

T_k – wektor chwil osiągnięcia wierzchołków należących do drogi dla k-tego obiektu,

$\tau^r(k)$ – chwila osiągnięcia wierzchołka $i^r(k)$ przez czoło k-tego obiektu,

$\underset{k=1,K}{\forall}\ \underset{r=0,R_k-1}{\forall}\tau^{r+1}(k) \geq \tau^r(k) \geq 0$, przy czym przyjmujemy, że $\underset{k=1,K}{\forall}\tau^0(k) \geq 0$ oraz

$$\tau^r(k) = \tau^0(k) + \sum_{j\in\{0,...,r-1\}} \frac{d_{i^j(k), i^{j+1}(k)}}{v_{i^j(k), i^{j+1}(k)}},\ r > 0, \tag{4}$$

V_k – wektor prędkości k-tego obiektu na łukach jego drogi,

$v_{i^r(k), i^{r+1}(k)}$ – prędkość k-tego obiektu na łuku $(i^r(k), i^{r+1}(k))$ między wierzchołkami $i^r(k)$ oraz $i^{r+1}(k)$ drogi,

R_k+1 – liczba wierzchołków drogi dla k-tego obiektu.

Niech $\Pi(s, t)$ opisuje zbiór wektorów $I(s, t) = (I_1, I_2, ..., I_K)$ dróg z $s = (s_1, s_2, ..., s_K)$ do $t = (t_1, t_2, ..., t_K)$. Zdefiniujmy τ^* jako najwcześniejszy moment dotarcia do wierzchołka docelowego najwolniejszego z K obiektów:

$$\tau^* = \min_{l(s,t)\in\Pi(s,t)} \max_{k\in\{1,...,K\}} \tau^{R_k}(k). \tag{5}$$

Oznaczmy ponadto przez IP_k następujący wektor punktów wyrównania:

$$IP_k = (i_1(k), i_2(k), ..., i_p(k), ..., i_{p_k}(k)), \tag{6}$$

gdzie $i_p(k)$ – p-ta składowa wektora IP_k opisująca wierzchołek (punkt) wyrównania, w którym musimy wyrównać czoło k-tego obiektu w odniesieniu do czół pozostałych obiektów, spełniająca warunki:

$$\begin{gathered} \underset{p\in\{1,...,P_k\}}{\forall}\ \underset{r\in\{1,...,R_k\}}{\exists} i_p(k) = i^r(k), \\ r_p(k) = r \in \{1, ..., R_k\} \Leftrightarrow i_p(k) = i^r(k). \end{gathered} \tag{7}$$

Warunek (7) zapewnia, że dla k-tego obiektu droga I_k musi przechodzić przez wierzchołki należące do IP_k. Oznaczmy przez analogię do IP_k:

$$TP_k = (\tau_1(k), \tau_2(k), ..., \tau_p(k), ..., \tau_{p_k}(k)) \tag{8}$$

jako wektor chwil osiągnięcia przez *k*-ty obiekt punktów wyrównania należących do IP_k, $\tau_p(k)$ oznacza chwilę osiągnięcia *p*-tego punktu wyrównania przez *k*-ty obiekt,

$$\tau_p(k) = \tau^0(k) + \sum_{r \in \{0,...,r_p(k)-1\}} \frac{d_{i^r(k),i^{r+1}(k)}}{v_{i^r(k),i^{r+1}(k)}} \tag{9}$$

oraz przyjmijmy, że dla każdego $k = 1, \ldots, K$ zachodzi: $\tau_0(k) = \tau^0(k)$. W celu uproszczenia oznaczeń przyjmujemy, że $t_{i^r(k),i^{r+1}(k)} = \frac{d_{i^r(k),i^{r+1}(k)}}{v_{i^r(k),i^{r+1}(k)}}$, tzn. jest to czas przemieszczania *k*-tego obiektu na łuku $(i^r(k), i^{r+1}(k))$ między wierzchołkami $i^r(k)$ oraz $i^{r+1}(k)$ jego drogi. Przyjmiemy również, że $\tau_p^{\max}$ oznacza chwilę dotarcia do *p*-tego punktu wyrównania najwolniejszego obiektu, tzn.

$$\tau_p^{\max} = \max_{k \in \{1,..,K\}} \tau_p(k). \tag{10}$$

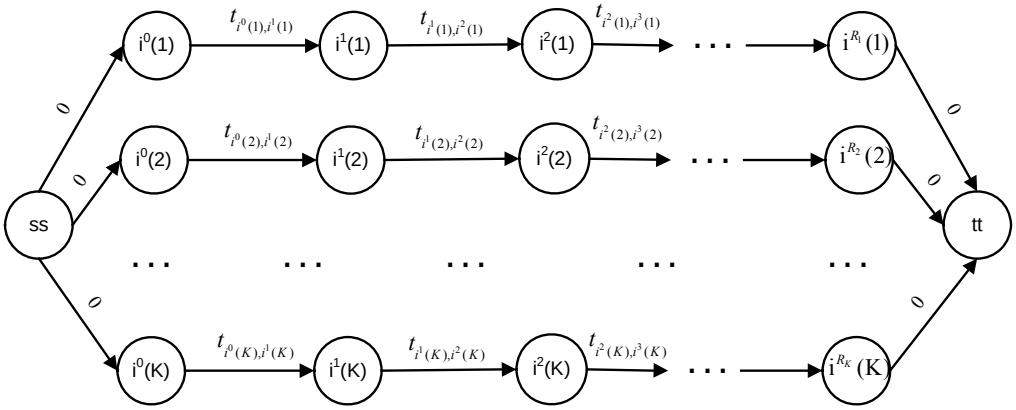
Dodatkowo zakładamy, że: $P_1 = P_2 = ... = P_K = N$, tzn. dla każdego obiektu dysponujemy taką samą liczbą punktów wyrównania. Niech k^* oznacza numer obiektu, dla którego wyznaczona droga jest najdłuższa, tzn. $k = k^* \Leftrightarrow \tau^{R_{k^*}}(k^*) = \max_{k \in \{1,...,K\}} \tau^{R_k}(k) = \tau^{\max}$. Zauważmy, że wówczas I_{k^*} oznaczać będzie ścieżkę krytyczną w sieci S^T, która opisuje pewne przedsięwzięcie. Zdarzeniami w tym przedsięwzięciu (wierzchołkami sieci) są elementy wektora I_k, czynności (łuki sieci) reprezentują odcinki drogi dla *k*-tego obiektu łączące kolejne wierzchołki $i^r(k)$ oraz $i^{r+1}(k)$, a na łukach opisana jest funkcja o wartościach równych czasom $t_{i^r(k),i^{r+1}(k)}$. Początkiem i końcem przedsięwzięcia są wirtualne wierzchołki, odpowiednio *ss*, *tt*, przy czym wierzchołek *ss* łączymy łukami z *K* wierzchołkami $i^0(\cdot)$, a wierzchołek *tt* – z *K* wierzchołkami $i^{R_k}(\cdot)$ i tym łukom (wirtualnym czynnościom) nadajemy czas realizacji równy zero (rys. 1).

W celu dalszych rozważań przyjmiemy następujące dodatkowe oznaczenia:

$$\Delta\tau_p^*(k) = \tau_p(k^*) - \tau_p(k), \tag{11}$$

$$\Delta\tau_p^* = \max_{k \in \{1,...,K\}} \Delta\tau_p^*(k), \tag{12}$$

$$FT(k) = \tau^{R_{k^*}}(k^*) - \tau^{R_k}(k). \tag{13}$$



Rys. 1. Przykład konstrukcji sieci S^T „spinającej" drogi dla K obiektów i interpretowanej jako sieć pewnego przedsięwzięcia, w którym zdarzeniami są wierzchołki wyrównania, a czynnościami odcinki dróg dla każdego obiektu

Używając określeń znanych z analizy ścieżki krytycznej, możemy zinterpretować $FT(k)$ jako luz czasowy dla k-tego obiektu, tzn. czas, o który łącznie mogą się opóźnić czasy realizacji na poszczególnych łukach drogi dla tego obiektu, aby nie spowodować zmiany czasu realizacji przedsięwzięcia (tzn. aby czas $\tau^{\max} = \max\limits_{k\in\{1,..,K\}} \tau^{R_k}(k)$ nie uległ zmianie).

3. Sformułowanie problemu harmonogramowania

Problem harmonogramowania zsynchronizowanego przemieszczania K obiektów definiujemy następująco [14]: wyznaczyć dla każdego obiektu $k \in \{1, ..., K\}$ drogę I_k przechodzącą przez punkty wyrównania będące składowymi wektora IP_k oraz dla każdego odcinka (łuku) $(i^r(k), i^{r+1}(k))$, $r \in \{0, ..., R_k - 1\}$ drogi taką prędkość $v_{i^r(k),i^{r+1}(k)}$, że:

$$\sum_{k=1}^{K} \tau^{R_k}(k) \to \min , \tag{14}$$

$$\sum_{k=1}^{K}\sum_{p=1}^{N} (\tau_p^{\max} - \tau_p(k)) \to \min , \tag{15}$$

przy ograniczeniach:

$$v_{i^r(k),i^{r+1}(k)} \le v^{\max}(k), \quad r=\overline{0,R_k-1}, \quad k=\overline{1,K}, \tag{16}$$

$$v_{i^r(k),i^{r+1}(k)} > 0, \quad r=\overline{0,R_k-1}, \quad k=\overline{1,K}, \tag{17}$$

gdzie $v^{\max}(k)$ oznacza maksymalną prędkość k-tego obiektu, wynikającą z jego możliwości technicznych. Przyjęcie ograniczenia (17) powoduje, że zabronione jest zatrzymywanie się na każdym łuku (odcinku drogi). Jeżeli (17) zapiszemy w postaci nieostrej nierówności, to dopuścimy zatrzymywanie się na tych odcinkach. Biorąc pod uwagę (9) i (10), funkcję celu (15) możemy zdefiniować następująco:

$$\sum_{p=1}^{N}\sum_{k=1}^{K}\left(\max_{l\in\{1,\ldots,K\}}\left(\tau^0(l)+\sum_{\substack{r\in\{0,\ldots,R_l-1\}\\ r\le r_p(l)}}\frac{d_{i^r(l),i^{r+1}(l)}}{v_{i^r(l),i^{r+1}(l)}}\right)-\left(\tau^0(k)+\sum_{\substack{r\in\{0,\ldots,R_l-1\}\\ r\le r_p(k)}}\frac{d_{i^r(l),i^{r+1}(l)}}{v_{i^r(l),i^{r+1}(l)}}\right)\right)\to\min. \tag{18}$$

Generalnie miary oceny harmonogramu możemy podzielić na dwa typy:

- pierwszego typu (C.1), definiujące miary oceny szybkości przemieszczania:

$$\tau^{\max}=\max_{k\in\{1,\ldots,K\}}\tau^{R_k}(k)\to\min, \tag{C.1.1}$$

$$\sum_{k=1}^{K}\tau^{R_k}(k)\to\min, \tag{C.1.2}$$

- drugiego typu (C.2), definiujące miary oceny „równoległości" przemieszczania:

$$\sum_{k=1}^{K}\sum_{p=1}^{N}(\tau_p^{\max}-\tau_p(k))\to\min, \tag{C.2.1}$$

$$\min_{p\in\{1,\ldots,N\}}\max_{k\in\{1,\ldots,K\}}(\tau_p^{\max}-\tau_p(k)), \tag{C.2.2}$$

$$\sum_{k=1}^{K}\sum_{p=1}^{N}|\tau_p^{avg}-\tau_p(k)|\to\min, \tag{C.2.3}$$

gdzie $\tau_p^{avg}=\frac{1}{K}\sum_{k=1}^{K}\tau_p(k)$.

Dodatkowo możemy rozpatrywać następujące rozszerzenia rozważanego problemu:
– dodając poniższe ograniczenie

$$\tau^0(k)+\sum_{r\in\{0,\ldots,R_k-1\}}\frac{d_{i^r(l),i^{r+1}(l)}}{v_{i^r(l),i^{r+1}(l)}}\le T^{\max}, \quad k=\overline{1,K}, \tag{19}$$

poszukujemy takiego harmonogramu przemieszczania, że moment osiągnięcia wierzchołka docelowego przez najwolniejszy obiekt jest nie większy niż pewna ustalona chwila $T^{\max} \geq \tau^*$;

– możemy osłabić warunek wynikający z funkcji celu (15), wprowadzając dodatkową grupę ograniczeń (nieliniowych)

$$|\tau_p^{\max} - \tau_p(k)| \leq \tau^{\text{delay}}, \quad k = \overline{1,K}, \quad p = \overline{1,N} \tag{20}$$

która gwarantuje, że wielkość opóźnienia (przyspieszenia) w każdym wierzchołku wyrównania dla każdego obiektu w stosunku do obiektu najwolniejszego będzie nie większa niż pewna ustalona wartość τ^{delay};

– możemy wprowadzić zmodyfikowany warunek (17) jako $v_{i^r(k),i^{r+1}(k)} \geq v^{\min}(k)$, gdzie $v^{\min}(k)$ jest minimalną dopuszczalną prędkością k-tego obiektu.

W pracy [14] pokazano, że problem (15)–(17) jest podobny do problemu szeregowania zadań przed liniami krytycznymi w celu minimalizacji sumy maksymalnych opóźnień w punktach (wierzchołkach) wyrównania [1], [7], ale występują zasadnicze różnice, które nie pozwalają skorzystać ze znanych algorytmów szeregowania: (a) zadania (odcinki dróg dla obiektów) są już przydzielone do konkretnych procesorów (przemieszczanych obiektów) i nie mamy na to wpływu, (b) decydujemy o opóźnieniach pracy procesorów w celu wydłużenia czasów realizacji zadań. Można powiedzieć, że rozpatrywany problem harmonogramowania jest w pewnym sensie problemem odwrotnym do problemu szeregowania zadań: zamiast przydzielać zadania do procesorów – zadania te już są arbitralnie przydzielone, za to decydujemy o opóźnieniach (czasach) pracy procesorów (a w zadaniu szeregowania czasy te były jednoznacznie podane). To zatem, co w problemie szeregowania zadań było parametrami, w naszym problemie staje się zmiennymi decyzyjnymi, a to co w problemie szeregowania zadań było zmiennymi decyzyjnymi – w naszym zadaniu stanowi parametry.

4. Algorytmy harmonogramowania zsynchronizowanego przemieszczania

4.1. Wprowadzenie

W pracy [14] zdefiniowano szczegółowo szereg modeli harmonogramowania oraz opisano ich własności. Założenia dodatkowe, które można brać pod uwagę przy sformułowaniu podstawowego problemu (14)–(17) mogą być następujące: drogi dla K obiektów muszą być rozłączne bądź nie, muszą przechodzić przez wskazane wcze-

śniej punkty wyrównania lub punkty te wyliczane są dynamicznie w trakcie przemieszczania, każdy k-ty obiekt ($k = 1, ..., K$) może planować przemieszczanie tylko wewnątrz pewnych podobszarów, itp. W pierwszym przypadku, gdy składowe wektorów s i t są różne (czyli dysponujemy K parami różnych wierzchołków początkowych i końcowych), mamy do czynienia z NP-trudnym problemem poszukiwania K dróg rozłącznych i możemy go rozwiązać, używając pewnych algorytmów przybliżonych [10], [11]. Jeżeli natomiast komponenty wektorów s i t są identyczne (czyli K obiektów startuje z tego samego wierzchołka początkowego i kończy w tym samym wierzchołku końcowym, przy czym $s \neq t$), to wówczas możemy skorzystać z wniosków wynikających z twierdzeń Halla i Mengera dotyczących dróg wierzchołkowo lub łukowo-rozłącznych i zastosować algorytm wyznaczania przepływu zaspokajającego o minimalnym koszcie w pewnej sieci zastępczej [10]. W drugim i trzecim przypadku możemy zastosować jeden ze znanych algorytmów wyznaczania dróg przechodzących przez wskazane wierzchołki sieci [4]. Pewien specyficzny przypadek problemu rozłącznych dróg opisuje sytuacja, kiedy każdy obiekt ma narzucony z góry swój „pas" terenu, w którym się przemieszcza, pasy są parami rozłączne i generując na bazie takich pasów podgrafy grafu G, a następnie poszukując w każdym z nich najkrótszej drogi, wyznaczamy faktycznie rozłączne drogi dla obiektów [12]. W pracy [14] pokazano, że jednym ze sposobów rozwiązania problemu (14)–(17) może być podejście dwuetapowe: najpierw wyznaczane są drogi najkrótsze I_k dla każdego k-tego obiektu (przy założeniu, że obiekty poruszają się w sieci z maksymalną prędkością $v^{\max}(k)$), czyli optymalizujemy ze względu na kryterium (14), a następnie rozwiązywany jest problem (15)–(17) polegający na takim poprawieniu (zmniejszeniu) prędkości odcinkowych dla każdego obiektu, aby uzyskać efekt „równoległości" przemieszczania obiektów, mierzony za pomocą wartości funkcji (15). Podejście to odpowiada wyznaczaniu rozwiązania leksykograficznego problemu dwukryterialnego (14)–(17). Taki sposób dwuetapowy rozwiązywania prezentowanego problemu dwukryterialnego i taka kolejność ważności kryteriów są dość intuicyjne: wyznaczamy zbiór (wektor) najkrótszych dróg dla K obiektów, aby ustalić optymalne trasy przemieszczania, przy założeniu, że poruszamy się z maksymalną możliwą prędkością na każdym łuku (otrzymamy zatem wektor najkrótszych dróg z najwcześniejszymi możliwymi terminami dojazdu do wszystkich wierzchołków pośrednich oraz końcowych), a w drugim etapie próbujemy poprawić (zmniejszyć) te prędkości, aby optymalizować ze względu na drugie kryterium. Zauważmy jednak, że zakładamy, iż dla każdego wektora dróg $I(s, t)$ z $s = (s_1, s_2, ..., s_K)$ do $t = (t_1, t_2, ..., t_K)$, dla którego wartość funkcji (14) jest minimalna, przyjęcie któregokolwiek z nich w drugim etapie jest tak samo dobre (tzn. wartość funkcji celu (15) jest minimalna dla każdego z nich). Tylko wówczas bowiem można otrzymać optymalne rozwiązanie zadania dwukryterialnego przy założonej wcześniej kolejności ważności kryteriów. W przeciwnym przypadku możemy przyjąć następujące postępowanie: wyznaczamy po kolei dla każdego k-tego obiektu n-tą najkrótszą drogę (tzn. pierwszą najkrótszą, drugą najkrótszą itd.) zgodnie z podejściem

opisanym np. w pracy [3], a następnie rozpatrując wszystkie możliwe kombinacje *n*-tych najkrótszych dróg dla *K* obiektów i takie, że wartość funkcji celu (15) jest minimalna, wyznaczymy rozwiązanie optymalne problemu dwukryterialnego. Podejście takie może być jednak nieefektywne czasowo.

W rozwiązaniach prezentowanych w niniejszym rozdziale zakładamy, że mamy wyznaczone najkrótsze drogi dla *K* obiektów. Można zatem powiedzieć, że abstrahujemy od sposobu wyznaczenia tych dróg, a skupiamy się na problemie „zrównoleglenia" przemieszczania. Problem, którym będziemy się zajmować i dla którego podamy dwa algorytmy rozwiązania sformułujemy zatem następująco: dla ustalonych dróg I_k dla każdego *k*-tego obiektu (wynik pierwszego etapu podejścia opisanego wyżej) wyznaczyć takie prędkości $v_{i^r(k),i^{r+1}(k)}$, $r=\overline{0,R_k-1}$, $k=\overline{1,K}$, że funkcja celu (15) przyjmuje wartość minimalną przy ograniczeniach (16)–(17). Niech $\tau'_p(k)$ oznacza zmodyfikowaną (na podstawie algorytmu) chwilę osiągnięcia przez *k*-ty obiekt *p*-tego punktu wyrównania oraz $\Delta\tau'_p(k)=\tau'_p(k)-\tau_p(k)$ oznacza czas, o który zmieni się zmodyfikowana (przez algorytm) chwila dotarcia *k*-tego obiektu do *p*-tego punktu wyrównania.

4.2. Algorytm programowania dynamicznego

Algorytm A.1 (minimalizacja sumy opóźnień w punktach wyrównania)

Dla każdego $k \in \{1, ..., K\}$ *wylicz zmodyfikowane terminy dotarcia do wierzchołków wyrównania:*

$$\begin{cases} \tau'_p(k)=\tau^0(k), & \text{dla } p=0, \\ \tau'_p(k)=\max\limits_{l\in\{1,\ldots,K\}}(\Delta\tau'_{p-1}(l)+\tau_p(l)), & \text{dla } 1\le p\le N. \end{cases} \tag{21}$$

Zauważmy, że $\forall\limits_{k\in\{1,\ldots,K\}}\Delta\tau_p(k)\ge 0,$ co wynika ze wzoru (21) oraz z założenia, że $\forall\limits_{k=\overline{1,K}}$ $\forall\limits_{r=\overline{1,R_k-1}}$ $\tau^{r+1}(k)\ge\tau^r(k)\ge 0$. Stąd wniosek, że $\tau'_p(k)\ge\tau_p(k)$ dla każdego *p* i *k*, zatem warunek (16) jest spełniony (przypomnijmy, że $\tau_p(k)$ jest najwcześniejszym możliwym terminem osiągnięcia *p*-tego wierzchołka wyrównania dla *k*-tego obiektu). Mając dane $\forall\limits_{p\in\{1,\ldots,N\}}\ \forall\limits_{k\in\{1,\ldots,K\}}\tau'_p(k)$ i $\Delta\tau'_p(k)$, możemy łatwo wyliczyć: $\forall\limits_{k\in\{1,\ldots,K\}}\ \forall\limits_{r\in\{0,\ldots,R_k\}}$ $\tau'^r(k):=\tau^r(k)+\Delta\tau'_{q(r)}(k)$, gdzie $q(r)=\max\{p\in\{1,\ldots,N\}: r_p(k)\le r\}$, oraz zmodyfikowane prędkości odcinkowe równe: $\forall\limits_{k\in\{1,\ldots,K\}}\ \forall\limits_{r\in\{0,\ldots,R_k-1\}}$ $v'_{i^r(k),i^{r+1}(k)}:=$

$\frac{d_{i^r(k),i^{r+1}(k)}}{\tau'^{r+1}(k)-\tau'^{r}(k)}$. Złożoność algorytmu A.1 wynosi $\Theta(KN)$. Idea działania algorytmu A.1 została przedstawiona na rysunku 2, a wartości pewnych charakterystyk podano w tabeli 1.

Tabela 1. Wartości $\tau_p(k)$ oraz $\Delta\tau_p^*(k)$ dla danych z rys. 2

k	$\tau_p(k)$				$\Delta\tau_p^*(k)$			
	$p=1$	$p=2$	$p=3$	$p=4$	$p=1$	$p=2$	$p=3$	$p=4$
3	2	13	16	17	5	–1	–2	–2
2	5	9	13	16	2	3	1	–1
1	7	12	14	15	0	0	0	0

Twierdzenie 1
Algorytm A.1 znajduje optymalne rozwiązanie problemu (15)–(17).

Dowód
Dla ustalonego $p \in \{1, ..., N\}$ zachodzi: $\tau'_p(1) = \tau'_p(2) = ... = \tau'_p(K)$, zatem $\sum_{k=1}^{K} \tau_p'^{\max} - \tau'_p(k) = 0$. Ponieważ warunek ten zachodzi dla każdego $p \in \{1, ..., N\}$, więc $\sum_{p=1}^{N}\sum_{k=1}^{K} \tau_p'^{\max} - \tau'_p(k) = 0$, co kończy dowód. <>

Zauważmy, że jeżeli dowolne dwa kolejne wierzchołki wyrównania p i $p+1$ są kolejnymi wierzchołkami z drogi dla k-tego obiektu, tzn. $r_p(k) = i^r(k) \Rightarrow r_{p+1}(k) = i^{r+1}(k)$, to wówczas z (21) wynika, że:

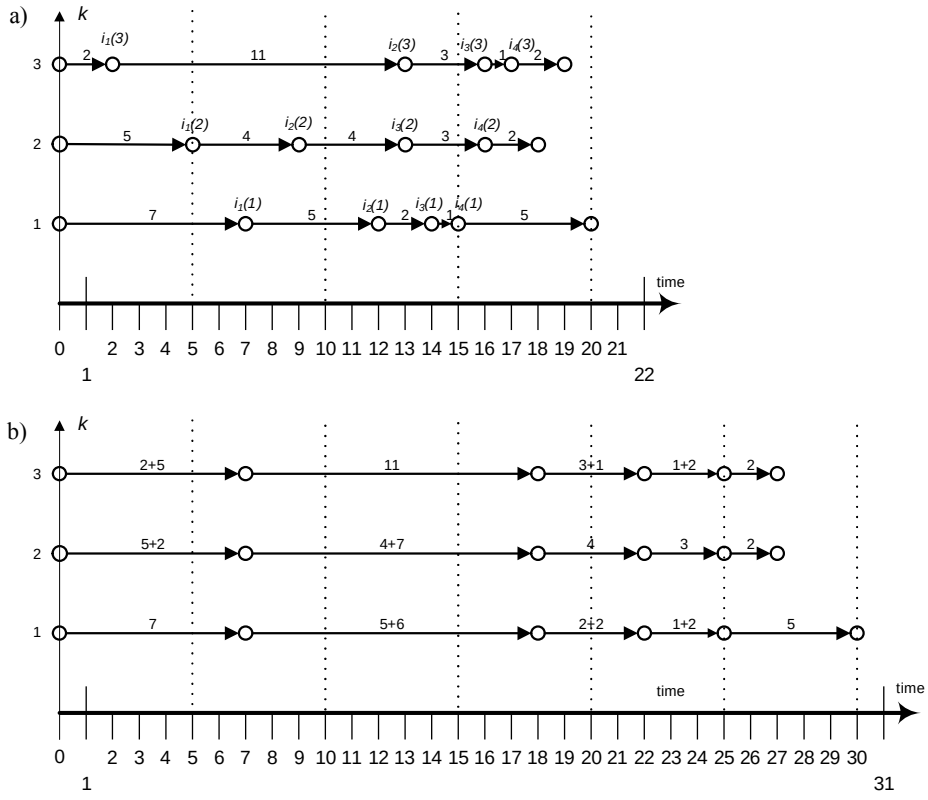
$$\begin{aligned}\tau'_p(k) &= \max_{l\in\{1,...,K\}}(\Delta\tau'_{p-1}(l)+\tau_p(l)) = \max_{l\in\{1,...,K\}}(\tau'_{p-1}(l)-\tau_{p-1}(l)+\tau_p(l))\\ &= \max_{l\in\{1,...,K\}}(\tau'_{p-1}(l)-\tau_{p-1}(l)+\tau_{p-1}(l)+t_{i^{r_{p-1}(l)}(l),i^{r_p(l)}(l)}) \qquad (22)\\ &= \max_{l\in\{1,...,K\}}(\tau'_{p-1}(l)+t_{i^{r_{p-1}(l)}(l),i^{r_p(l)}(l)}) = \tau'_{p-1}(k)+\max_{l\in\{1,...K\}} t_{i^{r_{p-1}(l)}(l),i^{r_p(l)}(l)}.\end{aligned}$$

Sytuację tę przedstawiono na rysunku 2. Na przykład dla $k = 2$ otrzymamy:

$$\tau'_2(2) = \tau'_1(2) + \max_{l\in\{1,...,3\}} t_{i^{r_1(l)}(l),i^{i_2(l)}(l)} = 7 + \max\{11, 4, 5\} = 18,$$

$$\tau'_3(2) = \tau'_2(2) + \max_{l\in\{1,...,3\}} t_{i^{r_2(l)}(l),i^{i_3(l)}(l)} = 18 + \max\{3, 4, 2\} = 22,$$

$$\tau'_4(2) = \tau'_3(2) + \max_{l\in\{1,...,3\}} t_{i^{r_3(l)}(l),i^{i_4(l)}(l)} = 22 + \max\{1, 3, 1\} = 25.$$



Rys. 2. Idea działania algorytmu A.1: (a) drogi dla $K = 3$ obiektów wraz z czasami osiągnięcia poszczególnych wierzchołków oraz z zaznaczonymi wierzchołkami wyrównania $i_p(k)$; (b) wynik działania algorytmu A.1 – czasy osiągnięcia poszczególnych wierzchołków wyrównania przez K obiekty są identyczne

Twierdzenie 2a

Warunkami koniecznymi do tego, aby dla każdego rozwiązania $\tau_N(k)$ otrzymanego z algorytmu A.1 zachodziło

$$\max_{k\in\{1,\ldots,K\}} \tau'_N(k) \le \tau_N(k^*) \tag{23}$$

są warunki:

1^o. $$\forall_{p\in\{1,\ldots,N\}} \; \forall_{k\in\{1,\ldots,K\}} \; \Delta\tau'_p(k) \ge 0\,, \tag{24}$$

2^o. $$\forall_{k\in\{1,\ldots,K\}} \; \Delta\tau^*_1(k) \le \Delta\tau^*_2(k) \le \ldots \le \Delta\tau^*_N(k)\,. \tag{25}$$

Dowód:

Ad.1° Załóżmy przeciwnie, tzn.

$$\exists_{p' \in \{1,\ldots,N\}} \; \exists_{k' \in \{1,\ldots,K\}} \Delta\tau'_{p'}(k') < 0 .$$

Wówczas z (11) wynika, że $\tau_{p'}(k^*) < \tau_{p'}(k')$. Ale z (21) wynika, że dla każdego $k \in \{1, \ldots, K\}$ prawdziwa jest równość: $\tau'_{p'}(k) = \max_{l \in \{1,\ldots,K\}} (\Delta\tau'_{p'-1}(l) + \tau_{p'}(l))$. Ponieważ zachodzi $\forall_{l \in \{1,\ldots,K\}} \Delta\tau_{p'-1}(l) \geq 0$, zatem $\tau'_{p'}(k') \geq \tau_{p'}(k') > \tau_{p'}(k^*)$. Jeżeli podstawimy $k' = k^*$ oraz $p' = N$, to otrzymamy $\tau'_N(k^*) > \tau_N(k^*)$. Sprzeczność ta kończy tę część dowodu.

Ad.2° Załóżmy przeciwnie, tzn.

$$\exists_{p' \in \{1,\ldots,N-1\}} \; \exists_{k' \in \{1,\ldots,K\}} \Delta\tau'_{p'}(k') > \Delta\tau'_{p'+1}(k') \tag{26}$$

oraz że spełnione są warunki wynikające z części 1° dowodu, tzn.

$$\forall_{k \in \{1,\ldots,K\}} \Delta\tau^*_{p'}(k') \geq 0, \quad \Delta\tau^*_{p'+1}(k) \geq 0 . \tag{27}$$

Wykażemy, że jeżeli zachodzi (26), to $\tau'_{p'+1}(k') > \tau_{p'+1}(k^*)$. Niech $p' = 1$. Wówczas z (26) mamy, że $\exists_{k' \in \{1,\ldots,K\}} \Delta\tau^*_1(k') > \Delta\tau^*_2(k')$ lub równoważnie

$$\exists_{k' \in \{1,\ldots,K\}} \tau_1(k') - \tau_1(k') > \tau_2(k^*) - \tau_2(k') . \tag{28}$$

Z (21) oraz z (27) wynika, że:

$$\tau'_1(k') = \max_{l \in \{1,\ldots,K\}} \tau_1(k) = \tau_1(k) , \tag{29}$$

$$\tau'_2(k') = \max_{k \in \{1,\ldots,K\}} (\tau'_1(k) - \tau_1(k) + \tau_2(k)) = \max_{k \in \{1,\ldots,K\}} (\tau_1(k) - \tau_1(k) + \tau_2(k)) . \tag{30}$$

Biorąc (28) mamy

$$\tau_1(k^*) - \tau_1(k') + \tau_2(k') - \tau_2(k^*) > 0 \tag{31}$$

lub równoważnie

$$\tau_1(k^*) - \tau_1(k') + \tau_2(k') > \tau_2(k^*) . \tag{32}$$

Wstawiając (30) do (31) otrzymujemy:

$$\tau'_2(k') = \max_{k \in \{1,\ldots,K\}} (\tau_1(k^*) - \tau_1(k) + \tau_2(k)) \geq \tau_1(k^*) - \tau_1(k') + \tau_2(k') > \tau_2(k^*) .$$

Jeżeli przyjmiemy, że $k' = k^*$, to mamy $\tau'_2(k^*) > \tau_2(k^*)$. Jak pamiętamy, I_{k*} oznacza ścieżkę krytyczną w sieci S^T, a otrzymana nierówność $\tau'_2(k^*) > \tau_2(k^*)$ oznacza, że wydłużeniu ulega czas realizacji ścieżki krytycznej I_{k*}, zatem i czas realizacji całego przedsięwzięcia, tzn. otrzymaliśmy sprzeczność z zależności (23). Wykazaliśmy zatem, że oprócz warunku (24) koniecznym do zachodzenia zależności (23) jest warunek (25). <>

Twierdzenie 2b

Warunki (24), (25) łącznie wystarczają do tego, aby dla każdego rozwiązania $\tau'_N(k)$ otrzymanego z algorytmu A.1 zachodziło (23).

Dowód

Aby dowieść tezy twierdzenia wykażemy, że jeżeli zachodzi (24) i (25), to wówczas dla każdego $p = 1, ..., N$

$$\tau'_p(k^*) \le \tau_p(k^*). \tag{33}$$

Dowód przeprowadzimy przez indukcję względem p. Z (29) wynika, że formuła (33) jest prawdziwa dla $p = 1$. Weźmy $p = m$ i załóżmy, że formuła (33) jest prawdziwa. Otrzymamy

$$\tau'_m(k^*) = \max_{l\in\{1,...,K\}} (\Delta\tau'_{m-1}(l) + \tau_m(l)) \le \tau_m(k^*). \tag{34}$$

Wykażemy teraz, że wyrażenie (33) jest prawdziwe dla $m + 1$. Mamy

$$\tau'_{m+1}(k^*) = \max_{l\in\{1,...,K\}} (\Delta\tau'_m(l) + \tau_{m+1}(l)) = \max_{l\in\{1,...,K\}} (\tau'_m(l) - \tau_m(l) + \tau_{m+1}(l)). \tag{35}$$

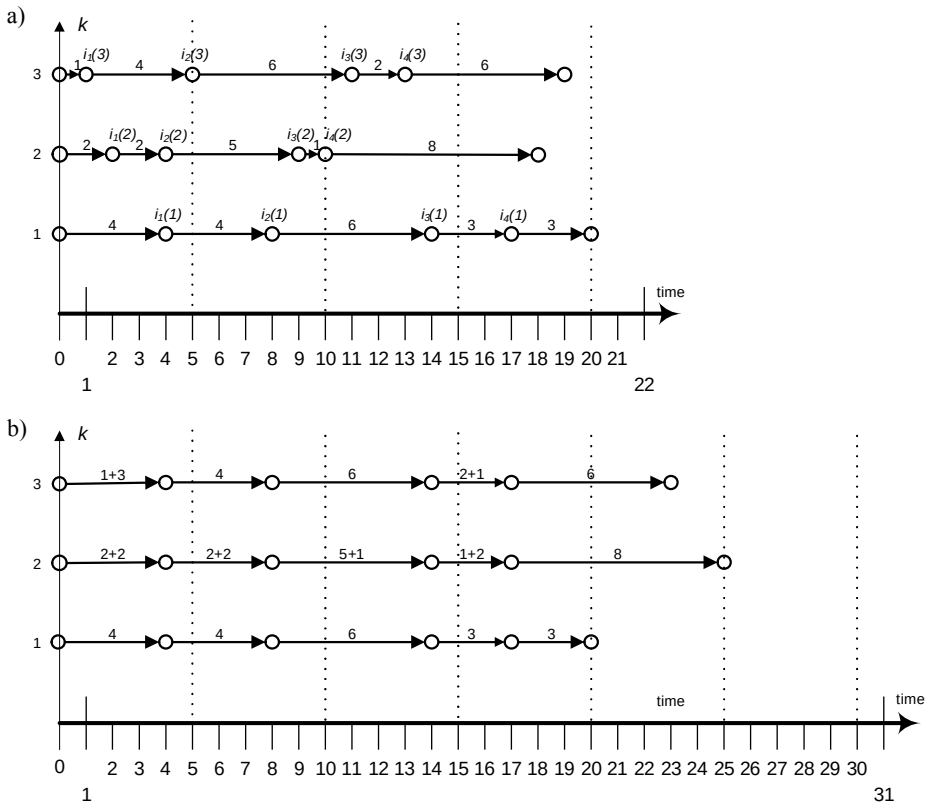
Z warunku (34) wynika, że $\tau'_m(k^*) \le \tau_m(k^*)$, z założenia (24), że $\forall_{l\in\{1,...,K\}} \tau_p(k^*) \ge \tau_p(l)$, a z założenia (25), że $\forall_{l\in\{1,...,K\}} \tau_m(k^*) - \tau_m(l) \le \tau_{m+1}(k^*) - \tau_{m+1}(l)$, zatem (35) możemy rozwinąć następująco:

$$\tau'_{m+1}(k^*) = \max_{l\in\{1,...,K\}} (\tau'_m(l) - \tau_m(l) + \tau_{m+1}(l)) \le \max_{l\in\{1,...,K\}} (\tau_m(k^*) - \tau_m(l) + \tau_{m+1}(l))$$
$$\le \max_{l\in\{1,...,K\}} (\tau_{m+1}(k^*) - \tau_{m+1}(l) + \tau_{m+1}(l)) \le \tau_{m+1}(k^*).$$

c.n.d. <>

Wniosek z twierdzenia 2b: jeżeli dla każdego $k = 1, \ldots, K$ ustalimy $r_N(k) = R_k$, to wówczas z zależności (23) mamy $\tau'^{R_k}(k) \le \tau^{R_{k^*}}(k^*)$, tzn. że najpóźniejszy moment zakończenia przemieszczania przez wszystkie obiekty nie ulegnie zmianie.

Na rysunku 3 przedstawiono wykorzystanie wniosków z twierdzenia 2a i 2b. Tabela 2 zawiera pewne charakterystyki problemu z rysunku 3.



Rys. 3. Drogi dla $K = 3$ obiektów spełniające warunki twierdzenia 2a i 2b:
a) wraz z czasami osiągnięcia poszczególnych wierzchołków oraz z zaznaczonymi wierzchołkami wyrównania $i_p(k)$; b) wynik działania algorytmu A.1 – czasy osiągnięcia poszczególnych wierzchołków wyrównania przez K obiekty są identyczne (równe 17)
i nie przekraczają czasów osiągnięcia dla obiektu $k^* = 1$

Tabela 2. Wartości $\tau_p(k)$ oraz $\Delta\tau_p^*(k)$ dla danych z rys. 3

k	$\tau_p(k)$				$\Delta\tau_p^*(k)$			
	$p=1$	$p=2$	$p=3$	$p=4$	$p=1$	$p=2$	$p=3$	$p=4$
3	1	5	11	13	3	3	3	4
2	2	4	9	10	2	4	5	7
1	4	8	14	17	0	0	0	0

Twierdzenie 3

Niech $\Delta\tau_p^d(k) = \max\{\Delta\tau_{p-1}^*(k) - \Delta\tau_p^*(k), 0\}$ oraz $\Delta\tau_p^d(k) = \max\limits_{k\in\{1,\ldots,K\}} \tau_p^d(k)$.

Jeżeli $\forall\limits_{p\in\{1,\ldots,N\}} \forall\limits_{k\in\{1,\ldots K\}} \Delta\tau_p^d(k) \geq 0,$ to

$$\forall_{p\in\{1,\ldots,N\}} \tau'_p(k^*) = \tau_p(k^*) + \sum_{\substack{p'\in\{1,\ldots,N\} \\ p'\leq p}} \Delta\tau_{p'}^d . \tag{36}$$

Dowód twierdzenia 3 zamieszczono w dodatku. Wnioski z twierdzenia 3:

1° $$\begin{aligned}\max_{k\in\{1,\ldots,K\}} \tau'^{R_k}(k) &= \tau'_N(k^*) + \max_{k\in\{1,\ldots,K\}} (\tau^{R_k}(k) - \tau_N(k)) \\ &= \tau_N(k^*) + \sum_{p\in\{1,\ldots,N\}} \Delta\tau_p^d + \max_{k\in\{1,\ldots,K\}} (\tau^{R_k}(k^*) - \tau_N(k)).\end{aligned}$$

2° Dla każdego

$$T^{\max} \geq \tau_N(k^*) + \sum_{p\in\{1,\ldots,N\}} \Delta\tau_p^d + \max_{k\in\{1,\ldots,K\}} (\tau^{R_k}(k) - \tau_N(k))$$

zachodzi

$$\sum_{p=1}^{N}\sum_{k=1}^{K} \tau_p'^{\max} - \tau'_p(k) = 0 .$$

Z wniosków 1° i 2° wynika, że dla każdego $T^{\max}$ spełniającego warunek 1° zachodzi: $\max\limits_{k\in\{1,\ldots,K\}} \tau'^{R_k}(k) \leq T^{\max}$, tzn. że najpóźniejszy moment zakończenia przemieszczania przez wszystkie obiekty nie przekroczy wartości $T^{\max}$ przy jednoczesnym spełnieniu warunku $\sum\limits_{p=1}^{N}\sum\limits_{k=1}^{K} \tau_p'^{\max} - \tau'_p(k) = 0$. Sprawdzenia warunku 2° możemy dokonać w czasie $\Theta(KN)$.

4.3. Algorytm oparty na analizie kosztowo-zyskowej

Aby rozwiązać problem (15)–(17) z dodatkowym ograniczeniem (19), w ogólności, zdefiniujemy ten problem w zmienionej formie: dla danego wektora dróg I_k dla każdego k-tego obiektu wyznaczyć takie $x_{k,p}$, $k = 1, \ldots, K$, $p = 1, \ldots, N$, że

$$\sum_{p=1}^{N}\sum_{k=1}^{K}\left(\max_{j\in\{1,\ldots,K\}}\left(\tau_p(j)+\sum_{i=1}^{p}x_{j,i}\right)-\left(\tau_p(k)+\sum_{i=1}^{p}x_{k,i}\right)\right)\to\min, \tag{37}$$

przy ograniczeniach:

$$\sum_{p=1}^{N}x_{k,p}\le FT(k),\qquad k=1,\ldots,K, \tag{38}$$

$$x_{k,p}\ge 0,,\quad k=1,\ldots,K;\quad p=1,\ldots,N, \tag{39}$$

gdzie $x_{k,p}$ oznacza czas, który jest dodawany do $\tau_p(k)$ dla k-tego obiektu w p-tym punkcie wyrównania. Zauważmy, że $\tau_p(k)+\sum_{i=1}^{p}x_{k,i}=\tau'_p(k)$. Dlatego też, jeśli oznaczymy przez $\Delta\tau'^{\max}_p(k)=\tau'^{\max}_p-\tau'_p(k)$, gdzie $\tau'^{\max}_p(k)=\max_{k\in\{1,\ldots K\}}\tau'_p(k)$, to wówczas funkcja (37) ma równoważną postać z $\sum_{p=1}^{N}\sum_{k=1}^{K}\Delta\tau'^{\max}_p\to\min$ i otrzymamy funkcję (15).

Zaprezentujemy obecnie heurystyczny algorytm A.*2* rozwiązujący problem (37)–(39), który jest równoważny z problemem (15)–(17) z dodatkowym warunkiem (19). Zdefiniujemy pewne dodatkowe oznaczenia wykorzystywane w algorytmie: card(x) – moc zbioru x; $a_p(k)$ – wartość czasu, która jest dodawana do $\tau_p(k)$. Zdefiniujemy również trzy następujące zbiory numerów punktów wyrównania spełniających pewne warunki:

$$P_s^{+}(k)=\{p\in\{s,\ldots,N\}:\Delta\tau'^{\max}_p(k)>0\}, \tag{40}$$

$$P_s^{\ge}(k)=\{p\in\{s,\ldots,N\}:\Delta\tau'^{\max}_p(k)-a_s(k)\ge 0\}, \tag{41}$$

$$P_s^{<}(k)=\{p\in\{s,\ldots,N\}:\Delta\tau'^{\max}_p(k)-a_s(k)<0\}. \tag{42}$$

Funkcje $Z(\cdot)$ i $L(\cdot)$ opisują „zysk" i „koszt" zwiększenia $\Delta\tau'^{\max}_p(k)$ o wartość $a_{s_k}(k)$, $s_k\in P^{+}_{s_k}(k)$:

$$Z(a_{s_k}(k))=a_{s_k}(k)\cdot\mathrm{card}(P^{\ge}_{s_k}(k))+\sum_{p\in P^{<}_{s_k}(k)}\Delta\tau'^{\max}_p(k), \tag{43}$$

$$L(a_{s_k}(k))=(K-1)\cdot\sum_{p\in P^{<}_{s_k}(k)}|\Delta\tau'^{\max}_p(k)-a_{s_k}(k)|. \tag{44}$$

Wartość $x_{k,p}:=x_{k,p}+a_p(k)$ (w kroku 10) jest równa sumie wartości $a_p(k)$ wyznaczonych we wszystkich iteracjach algorytmu dla każdego k i p. Idea algorytmu A.2 opiera

się na zmniejszaniu wartości $S = \sum_{p=1}^{N}\sum_{k=1}^{K}\Delta\tau_p'^{\max}(k)$ poprzez zmniejszenie wartości $\Delta\tau_p'^{\max}(k)$ dla pewnego k i p. Wartość sumy S zostanie zmniejszona wówczas, gdy wybierzemy taką największą wartość $a_{s_k}(k) \in (0, \min\{\Delta\tau_{s_k}'^{\max}(k), FT(k)\}]$ dla jakiegoś $s_k \in P_1^+(k)$, że $Z(a_{s_k}(k)) > L(a_{s_k}(k))$. Własność ta powoduje, że algorytm ten możemy zaliczyć do algorytmów zachłannych. Algorytm próbuje zmniejszyć wartość S aż do momentu, gdy $FT(k)$ dla wszystkich k będzie równe zero lub gdy a_{sk} nie będzie większe od zera (dla tych $a_{s_k}(k)$, dla których $Z(a_{s_k}(k)) > L(a_{s_k}(k))$) dla żadnego k i p (zmienna *licznik* > 0).

Przeanalizujmy działanie algorytmu A.2. Weźmy pod uwagę wiersz tabeli 4 dla k = 2. Opłaca się wybrać $a_1 = 2 = \min\{2, 2\}$, ponieważ kiedy zmniejszymy wartość $\Delta\tau_p'^{\max}(2)$ dla $p \in P_1^{\geq}(2) = 1, 2, 3\}$, wówczas nasz „zysk" (zmniejszenie wartości sumy S) jest równy (patrz tabela 5, wiersz = 3, kolumna = 4): $Z(a_1(2)) = a_1(2)\ \mathrm{card}(P_1^{\geq}(2)) + \sum_{p\in P_1^{<}(2)} \Delta\tau_p'^{\max}(2) = 2\cdot 3 + 1 = 7$. „Koszt" (zwiększenie wartości S) jest równy $L(a_1(2)) = (3-1) + \sum_{p\in P_1^{<}(2)} |\Delta\tau_p'^{\max}(2) - a_1(2)| = 2\cdot 1$. Następnie, w krokach 7÷9 zmniejszamy wartość $\Delta\tau_p'^{\max}(k)$ i $FT(k)$ o $a_{s_k}(k)$ dla wszystkich $p \geq s_k$. Otrzymamy: $\Delta\tau_1'^{\max}(2) = 2 - 2 = 0$, $\Delta\tau_2'^{\max}(2) = 4 - 2 = 2$, $\Delta\tau_3'^{\max}(2) = 3 - 2 = 1$, $\Delta\tau_4'^{\max}(2)$ = 1 – 2 = –1. Zauważmy, że ujemna wartość $\Delta\tau_4'^{\max}(2)$ sugeruje, że dla p = 4 zmianie ulega $\tau_4'^{\max}$. Początkowe wartości czasów osiągnięcia punktu wyrównania numer p = 4 były równe: $\tau_4'(1) = 17$, $\tau_4'(2) = 16$, $\tau_4'(3) = 15$, zatem $\tau_4'^{\max}$ = 17. Wynikają z tego różnice: $\Delta\tau_4'^{\max}(1)$ = 17 – 17 = 0, $\Delta\tau_4'^{\max}(2)$= 17 – 16 = 1, $\Delta\tau_4'^{\max}(3)$ =17 – 15 = 2. Ponieważ wartość czasu, którą dodano do drugiego obiektu (tzn. k = 2), dla każdego p jest równa $a_p(k)$ = 2, zatem nowe wartości $\tau_p'(2)$ są równe: $\tau_1'(2)$ = 7, $\tau_2'(2)$ = 11, $\tau_3'(2)$ = 15, $\tau_4'(2)$ = 18, czyli $\tau_4'^{\max}$ = 18 oraz $\Delta\tau_4'^{\max}(2)$ = 17 – 18 = –1. Ponieważ z definicji $\Delta\tau_p'(k)$ nie może być ujemne, więc dla p = 4 musimy zwiększyć wartość $\Delta\tau_4'(k)$, dla każdego k, o $|\Delta\tau_4'(2)| = 1$, stąd $\Delta\tau_4'^{\max}(1)$ = 0+|–1| = 1, $\Delta\tau_4'^{\max}(2)$ = –1+|–1| = 0, $\Delta\tau_4'^{\max}(3)$ = 2+|–1| = 3. Operacja ta realizowana jest w kroku 8 algorytmu wówczas, gdy $\Delta\tau_p'^{\max}(k) - a_{s_k}(k) < 0$ w kroku 7.

Algorytm A.2

Dane: I_k, T_k, IP_k, TP_k dla każdego $k=1,\ldots,K$;

Inicjalizacja: $\forall_{k\in\{1,\ldots,K\}}\ \forall_{p\in\{1,\ldots,N\}}\ a_p(k) := 0$; $\forall_{k\in\{1,\ldots,K\}}\ \forall_{p\in\{1,\ldots,N\}}\ x_{k,p} := 0$; $\forall_{k\in\{1,\ldots,K\}}\ FT(k) := \tau^{R_{k^*}}(k^*) - 0\tau^{R_k}(k)$;

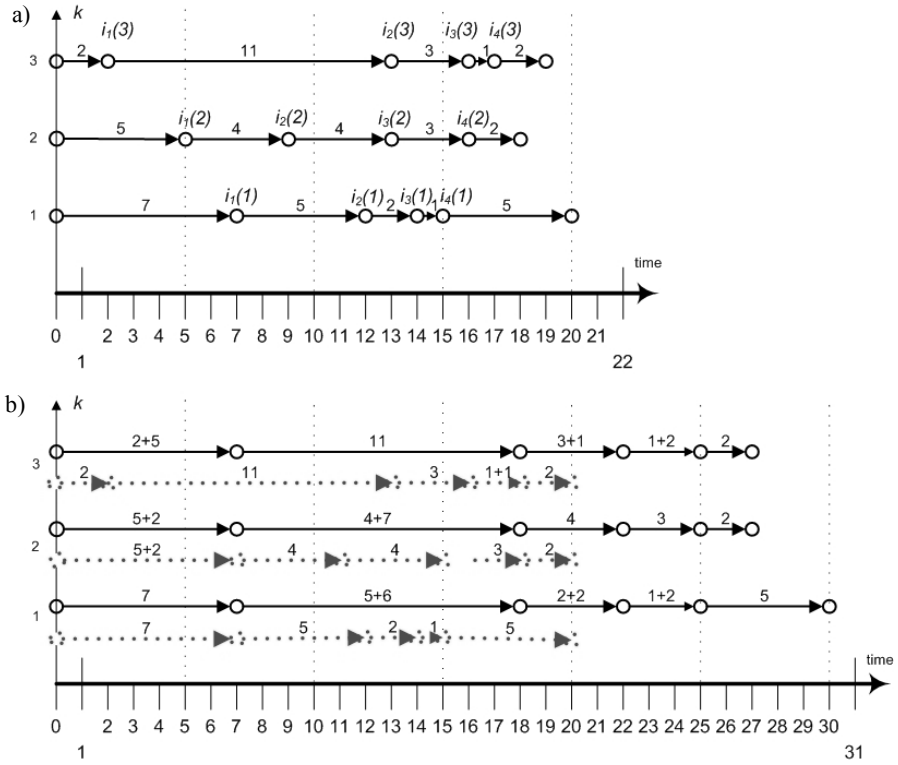
$\forall_{k\in\{1,\ldots,K\}}\ \forall_{p\in\{1,\ldots,N\}}\ \Delta\tau'^{\max}_p(k) := \tau^{\max}_p - \tau_p(k)$; licznik:= N;

1. WHILE $(\exists_{k\in\{1,\ldots,K\}} > 0) \wedge$ (licznik>0) DO
2. licznik:=0;
3. FOR $k=1,\ldots,K$ DO
4. IF $FT(k)>0$ THEN
5. Znajdź takie $s_k \in P_1^+(k)$ i maksymalną wartość $a_{s_k}(k) \in (0, \min\{\Delta\tau'^{\max}_{s_k}(k)$ $FT(k)\}]$, dla których warunek $Z(a_{s_k}(k)) > L(a_{s_k}(k))$ jest spełniony;
6. IF $a_{s_k}(k) > 0$ THEN
7. $\forall_{p\in\{1,\ldots,N\}}\ \Delta\tau'^{\max}_p(k) := \Delta\tau'^{\max}_p(k) - a_{s_k}(k)$;
8. $\forall_{p\in P^c_{s_k}(k)}\ \forall_{j\in\{1,\ldots,K\}}\ \Delta\tau'^{\max}_p(j) := \Delta\tau'^{\max}_p(j) + |\Delta\tau'^{\max}_p(k)|$;
9. $FT(k) := FT(k) - a_{s_k}(k)$;
10. $x_{k,s_k} := x_{k,s_k} + a_{s_k}(k)$;
11. licznik:=licznik+1;
12. END IF;
13. END IF;
14. END FOR;
15. END WHILE.

Oszacujemy obecnie złożoność algorytmu A.2. Liczba iteracji L_{WHILE} zewnętrznej pętli WHILE jest ograniczona od góry przez wartość: $L_{\text{WHILE}} < \max_{k\in\{1,\ldots K\}} \left\lceil \frac{FT(k)}{\Delta\tau^{\min}(k)} \right\rceil$, gdzie $\Delta\tau^{\min}(k) = \min\{1, \min\{\tau^{\min}_p - \tau_p(k): p \in \{1, ..., N\},\ \tau^{\max}_p - \tau_p(k) > 0\}\}$. Łatwo zauważyć, że złożoność poszczególnych kroków algorytmu wynosi: dla kroku 5 – $O(N^2)$, dla kroku 7 – $O(N)$, dla kroku 8 – $O(KN)$, a dla kroków 9 i 10 – $O(1)$. Ponieważ kroki 4÷14 wykonują się w pętli K razy, złożoność algorytmu wynosi więc $O(L_{\text{WHILE}}\,(K^2N + KN^2))$.

4.4. Przykład liczbowy

Zaprezentowany na rysunku 4 przykład pokazuje porównanie wyników, otrzymanych z realizacji algorytmu A.1 i A.2, dla $K = 3$ obiektów, dla których dysponujemy $N = 4$ punktami wyrównania.



Rys. 4. a) Wektor dróg dla $K = 3$ obiektów z pożądanymi czasami osiągnięcia każdego z $N = 4$ punktów wyrównania dla każdego obiektu (tab. 3).
b) Wyniki realizacji algorytmu A.1 (linia ciągła) oraz A.2 (linia przerywana)

Tabela 3. Wartości funkcji $\tau_p(k)$ i $\tau^{R_k}(k)$

k	p				$\tau^{R_k}(k)$
	1	2	3	4	
3	2	13	16	17	19
2	5	9	13	16	18
1	7	12	14	15	20

Tabela 4. Wartości początkowe funkcji $\Delta\tau_p'^{\max}(k)$ oraz $FT(k)$ (przed uruchomieniem algorytmu A.2)

k	p				$FT(k)$
	1	2	3	4	
3	5	0	0	0	1
2	2	4	3	1	2
1	0	1	2	2	0

Tabela 5. Wyniki uruchomienia wszystkich iteracji (2) algorytmu A.2

Nr iteracji	k	s_k	$a_{s_k}(k)$	$Z(a_{s_k}(k))$	$L(a_{s_k}(k))$
1	1	0	0	0	0
	2	0	2	7	2
	3	0	0	0	0
2	1	4	1	1	0
	2	0	0	0	0
	3	0	0	0	0

Tabela 6. Wartości końcowe funkcji $\Delta\tau_p'^{\max}(k)$ oraz $FT(k)$ (po uruchomieniu algorytmu A.2)

k	p				$FT(k)$
	1	2	3	4	
3	5	0	0	0	0
2	0	2	1	0	0
1	0	1	2	3	0

Tabela 7. Wartości końcowe $x_{k,p}$ (po uruchomieniu algorytmu A.2)

k	p			
	1	2	3	4
3	0	0	0	1
2	2	0	0	0
1	0	0	0	0

Tabela 8. Zmodyfikowane terminy $\tau_p'(k)$ dotarcia poszczególnych obiektów do punktów wyrównania (po uruchomieniu algorytmu A.2)

k	p			
	1	2	3	4
3	2	13	16	17+1
2	5+2	9+2	13+2	16+2
1	7	12	14	15

W tabeli 4 podano wartości początkowe funkcji $\Delta\tau_p'^{\max}(k)$ oraz $FT(k)$ przed uruchomieniem algorytmu A.2. Tabela 6 zawiera końcowe wartości tych funkcji, po zakończeniu działania algorytmu A.2. W tabeli 7 podano wyliczone wartości zmiennych $x_{k,p}$ na podstawie sumy wartości $a_p(k)$, wyznaczonych we wszystkich iteracjach algorytmu dla każdego k i p (patrz tab. 5). Biorąc pod uwagę wartości zapisane w tabeli 7

oraz zależność $\tau'_p(k) = \tau_p(k) + \sum_{i=1}^{p} x_{k,j}$, otrzymamy zmodyfikowane terminy dotarcia obiektów do punktów wyrównania zawarte w tabeli 8. Korzystając z rozważań zawartych w rozdziale 4.2, mając zmodyfikowane terminy $\tau'_p(k)$ dotarcia obiektów do punktów wyrównania oraz znając odległości geograficzne $d_{i^r(k),i^{r+1}(k)}$ między punktami wyrównania $i_r(k)$, $i_{r+1}(k)$, możemy wyliczyć zmodyfikowane prędkości odcinkowe $v'_{i^r(k),j^{r+1}(k)}$ następująco: $\underset{k\in\{1,\ldots,K\}}{\forall} \underset{r\in\{0,\ldots,R_k-1\}}{\forall} v'_{i^r(k),j^{r+1}(k)} := \dfrac{d_{i^r(k),i^{r+1}(k)}}{\tau'^{r+1}(k) - \tau'^{r}(k)}$.

5. Wnioski

Prezentowane w rozdziale 4 algorytmy wykorzystywane są w module harmonogramowania zsynchronizowanego przemieszczania wielu obiektów, stosowanym w pewnym systemie symulacyjnego wspomagania szkolenia operacyjnego sztabów [13]. Ponieważ są one bardzo szybkie z punktu widzenia oszacowanej złożoności obliczeniowej (co jest istotne z punktu widzenia szybkości reakcji przez symulator na interakcję użytkownika), spełniają swoje zadanie. Przeprowadzenie szerokich badań efektywnościowych obu algorytmów może być przedmiotem dalszych analiz. Ważnym elementem rozważań jest to, iż algorytm A.2 jest algorytmem heurystycznym, bazującym na strategii zachłannej. Można pokazać, że wartość rozwiązania otrzymywanego z algorytmu zależy od kolejności rozpatrywania obiektów w kolejnych iteracjach pętli zewnętrznej. Istotne jest zatem określenie wpływu kolejności rozpatrywania obiektów na wartość rozwiązania, które może być przedmiotem dalszych rozważań, jak również porównanie wartości rozwiązania otrzymywanego z algorytmów A1, A2 oraz poprzez zastosowanie klasycznych metod rozwiązywania sformułowanych zadań optymalizacji.

Zauważmy, że na bazie zdefiniowanych problemów (rozdział 3) oraz algorytmów z nimi związanych (rozdział 4) można rozważać wiele innych problemów. Jednym z nich jest problem następujący: równoległość definiowana jest w ten sposób, że w żadnym wierzchołku wyrównania ani opóźnienie, ani przyspieszenie wzajemne wszystkich jednostek nie może przekraczać ΔT, a kryterium oceny jest suma czasów dotarcia do celu wszystkich obiektów:

$$\sum_{k=1}^{K} \left(\tau_N(k) + \sum_{i=1}^{N} x_{k,i} \right) \to \min ,$$

przy ograniczeniach:

$$\sum_{k=1}^{K}\left(\max_{j\in\{1,\ldots,K\}}\left(\tau_p(j)+\sum_{i=1}^{p}x_{j,i}\right)-\left(\tau_p(k)+\sum_{i=1}^{p}x_{k,i}\right)\right)\le\Delta T,\quad p=1,\ldots,N,$$

$$\sum_{p=1}^{N}x_{k,p}\le FT(k),\quad k=1,\ldots,K,$$

$$x_{k,p}\ge 0,\quad k=1,\ldots,K,\quad p=1,\ldots,N.$$

Ten i inne problemy poruszane w pracy oraz w poprzedniej pracy autora [14] mogą być przedmiotem dalszych rozważań, zwłaszcza w kontekście zaadaptowania prezentowanych (lub skonstruowania nowych) algorytmów do ich rozwiązania.

Bibliografia

[1] BLAZEWICZ J., ECKER K.H., PESCH E., SCHMIDT G., WEGLARZ J., *Scheduling Computer and Manufacturing Processes*, Springer-Verlag, Heidelberg–Berlin-New York 2001.

[2] BUCHLI J. (ed.), *Mobile Robotics Moving Intelligence*, ISBN 3-86611-284-X, Pro Literatur Verlag, Germany, 2006.

[3] EPPSTEIN D., *Finding the K shortest Paths*, SIAM J. Computing, 1999, 28(2), 652–673.

[4] IBARAKI T., *Algorithms for obtaining shortest paths visiting specified nodes*, SIAM Review, 1973, Vol. 15, No. 2, Part 1, 309–317.

[5] LOGAN B., *Route planning with ordered constraints*, Proceedings of the 16th Workshop of the UK Planning and Scheduling Special Interest Group, December, Durham (UK), 1997.

[6] LONGTIN M., MEGHERBI D., *Concealed routes in ModSAF* [in:] Proceedings of the 5th Conference on Computer Generated Forces and Behavioural Representation, Technical Report, Institute for Simulation and Training, 1995, 305–314.

[7] LEUNG J.Y.-T. (ed.), *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, Chapman & Hall/CRC, Boca Raton–London–New York–Washington 2004.

[8] PETTY M.D., *Computer generated forces in Distributed Interactive Simulation*, Proceedings of the Conference on Distributed Interactive Simulation Systems for Simulation and Training in the Aerospace Environment, 19–20 April, Orlando (USA) (1995), 251–280.

[9] RAJPUT S., KARR C., *Unit Route Planning*, Technical Report IST-TR-94-42, Institute for Simulation and Training, Orlando (USA), 1994.

[10] SCHRIJVER A., SEYMOUR P., *Disjoint paths in a planar graph – a general theorem*, SIAM Journal of Discrete Mathematics, 1992, 5, 112–116.

[11] SCHRIJVER A., *Combinatorial Optimization. Polyhedra and Efficiency*, Springer-Verlag, Berlin–New York 2004.

[12] TARAPATA Z., *Modelling, optimisation and simulation of groups movement according to group pattern in multiresolution terrain-based grid network*, Proceedings of the 3rd NATO Regional Conference on Military Communication and Information Systems, 10–12 October, Zegrze (Poland), 2001, Vol. I, 241–251.

[13] TARAPATA Z., *Synchronization method of many objects movement in Computer Generated Forces systems*, Proceedings of the 7th NATO Regional Conference on Military Communication and Information Systems, 04–05 October, Zegrze (Poland), 2005, 93–99.

[14] TARAPATA Z., *Modele harmonogramowania zsynchronizowanego przemieszczania wielu obiektów*, Badania Operacyjne i Decyzje, 2007, nr 2, 83–103.

DODATEK

Dowód twierdzenia 3

Zauważmy najpierw, że jeżeli $\forall_{k\in\{1,\ldots,K\}} \Delta\tau_1^*(k) \le \Delta\tau_2^*(k) \le \ldots \le \Delta\tau_N^*(k)$, tzn. spełniony jest warunek (25), to wówczas zachodzi

$$\forall_{p\in\{1,\ldots,N\}} \Delta\tau_p^d(k) = \max\{\Delta\tau_{p-1}^*(k) - \Delta\tau_p^*(k), 0\} = 0 \rightarrow \Delta\tau_p^d = 0,$$

stąd $\tau_p'(k^*) = \tau_p(k^*)$.

Aby wykazać prawdziwość (36) pokażemy, korzystając z (21), że $\forall_{p\in\{1,\ldots,N\}}$ zachodzi:

$$\tau_p'(k^*) = \tau_p(k^*) + \sum_{\substack{p'\in\{1,\ldots N\}\\ p'\le p}} \Delta\tau_{p'}^d = \max_{k\in\{1,\ldots K\}} (\Delta\tau_{p-1}'(k) + \tau_p(k)). \tag{45}$$

Wykażemy, poprzez indukcję względem p, że lewa strona $L = \tau_p(k^*) + \sum_{\substack{p'\in\{1,\ldots N\}\\ p'\le p}} \Delta\tau_{p'}^d$ wyrażenia (45) równa jest prawej $R = \max_{k\in\{1,\ldots K\}} (\Delta\tau_{p-1}'(k) + \tau_p(k))$, tzn. $L = R$.

Jeśli $p = 1$, to $L = \tau_1(k^*)$, a z $\Delta\tau_p^d(k) = \max\{\Delta\tau_{p-1}^*(k) - \Delta\tau_p^*(k), 0\}$ i $\Delta\tau_p^d(k) = \max_{k\in\{1,\ldots K\}} \Delta\tau_p^d(k)$ oraz (11) wynika, że $\Delta\tau_p^d(k) = \max\{\Delta\tau_0^*(k) - \Delta\tau_1^*(k), 0\}$ $= 0 \rightarrow \Delta\tau_1^d = 0$, czyli $L = \tau_1(k^*)$.

Prawa strona wyrażenia (45) wynosi

$$R = \max_{k\in\{1,\ldots K\}} (\Delta\tau_0'(k) + \tau_1(k)) = \max_{k\in\{1,\ldots K\}} (\tau_0'(k) - \tau_0(k) + \tau_1(k)) = \max_{k\in\{1,\ldots K\}} \tau_1(k) = \tau_1(k^*).$$

Otrzymaliśmy zatem $L = R$.

Jeśli $p = 2$, to mamy:

$$\begin{aligned} L &= \tau_2(k^*) + \Delta\tau_1^d + \Delta\tau_2^d = \tau_2(k^*) + \max_{k\in\{1,\ldots K\}} \Delta\tau_2^d(k) \\ &= \tau_2(k^*) + \max_{k\in\{1,\ldots K\}} (\max(\Delta\tau_1^*(k) - \Delta\tau_2^*(k), 0\}) \\ &= \max_{k\in\{1,\ldots K\}} (\max(\Delta\tau_1^*(k) - \Delta\tau_2^*(k), 0\} + \tau_2^*(k)), \end{aligned}$$

$$\begin{aligned} R &= \max_{k\in\{1,\ldots K\}} (\Delta\tau_1'(k) + \tau_2(k)) = \max_{k\in\{1,\ldots K\}} (\tau_1'(k) - \tau_1(k) + \tau_2(k)) \\ &= \max_{k\in\{1,\ldots K\}} (\tau_1(k) - \tau_1(k) + \tau_2(k)) = \max_{k\in\{1,\ldots K\}} (\Delta\tau_1^*(k) + \tau_2(k)). \end{aligned}$$

Z analizy *L* i *R* otrzymujemy, że aby zachodziło *L* = *R*, musi zachodzić, że $\exists_{l_1,l_2\in\{1,\ldots,K\}}$ takie, że

$$\max\{\Delta\tau_1^*(l_1)-\Delta\tau_2^*(l_2),0\}+\tau_2(k^*)=\Delta\tau_1^*(l_1)+\tau_2(l_2),$$

czyli

$$\max\{\tau_1(k^*)-\tau_1(l_1)-\tau_2(l_1)+\tau_2(l_1)+\tau_2(k^*),\tau_2(k^*)\}=\Delta\tau_1^*(l_2)+\tau_2(l_2).$$

Stąd

$$\max\{\tau_1(k^*)-\tau_1(l_1)+\tau_2(l_1)+\tau_2(k^*)\}=\tau_1(k^*)-\tau_1(l_2)+\tau_2(l_2). \tag{46}$$

Równość (46) jest zawsze prawdziwa, bowiem $\tau_1(k^*)-\tau_1(l_1)\geq 0$ (założenie twierdzenia), zatem $\tau_1(k^*)-\tau_1(l_1)+\tau_2(l_1)\geq\tau_2(k^*)$ i zachodzi *L* = *R*, dla $l_1 = l_2$.

Przyjmijmy, że $p = m = 2$ i wykażemy, że wyrażenie (36) jest prawdziwe dla $m + 1$. Zauważmy, że

$$L=\tau'_p(k^*)=\tau_p(k^*)+\sum_{\substack{p'\in\{1,\ldots,N\}\\ p'\leq p}}\Delta\tau_{p'}^d=\tau'_{p-1}(k^*)+\Delta\tau_p^d-\tau_{p-1}(k^*)+\tau_p(k^*).$$

Dla $m + 1$ otrzymamy zatem:

$$L=\tau'_m(k^*)+\Delta\tau_{m+1}^d-\tau_m(k^*)+\tau_{m+1}(k^*),$$

$$R=\max_{k\in\{1,\ldots,K\}}(\Delta\tau'_m(k)+\tau_{m+1}(k))$$

i dalej

$$L=\tau'_m(k^*)+\max_{k\in\{1,\ldots,K\}}(\max\{\Delta\tau_m^*(k)-\Delta\tau_{m+1}^*(k),0\}-\tau_m(k^*)+\tau_{m+1}(k^*)$$

$$=\max_{k\in\{1,\ldots,K\}}\left(\max\begin{Bmatrix}\Delta\tau_m^*(k)-\Delta\tau_{m+1}^*(k)-\tau_m(k^*)+\tau_{m+1}(k^*)+\tau'_m(k),\\ -\tau_m(k^*)+\tau_{m+1}(k^*)+\tau'_m(k)\end{Bmatrix}\right).$$

Aby zachodziło *L* = *R*, musi zachodzić, że $\exists_{l_1,l_2\in\{1,\ldots,K\}}$ takie, że

$$\max\begin{Bmatrix}\Delta\tau_m^*(k)-\Delta\tau_{m+1}^*(k)-\tau_m(k^*)+\tau_{m+1}(k^*)+\tau'_m(k),\\ -\tau_m(k^*)+\tau_{m+1}(k^*)+\tau'_m(k)\end{Bmatrix}=\Delta\tau'_m(l_2)+\tau_{m+1}(l_2),$$

czyli

$$\max\begin{Bmatrix}\tau_m(k)-\tau_m(l_1)-\tau_{m+1}(k^*)+\tau_{m+1}(l_1)-\tau_m(k^*)+\tau'_m(k^*),\\ -\tau_m(k^*)+\tau_{m+1}(k^*)+\tau'_m(k)\end{Bmatrix}=\Delta\tau'_m(l_2)+\tau_{m+1}(l_2),$$

co po uproszczeniu daje warunek:

$$\begin{aligned}\max\{\tau_{m+1}(l_1)-\tau_m(l_1)+\tau'_m(k^*),-\tau_m(k^*)+\tau_{m+1}(k^*)+\tau_m(k^*)\}\\ =\tau_{m+1}(l_2)-\tau_m(l_2)+\tau'_m(l_2).\end{aligned} \tag{47}$$

Jeżeli przyjmiemy, że $l_1 = l_2 = k^*$, to równość (47) zachodzi. Zachodzi również dla dowolnego $l_1 = l_2$ pod warunkiem, że $\tau_{m+1}(l_1) - \tau_m(l_1) \geq \tau_{m+1}(k^*) - \tau_m(k^*)$. <>

Scheduling algorithms of synchronized movement of many objects

The paper presents algorithms of determining a synchronized movement schedule of many objects. The author defines movement scheduling as a bicriterion nonlinear optimization problem. He also presents a method of solving the bicriterion problem which is based on finding a lexicographic solution. Two scheduling algorithms of synchronized movement and their properties are given. The first algorithm is based on the dynamic programming, the second one is based on the cost-profit analysis. The necessary and sufficient conditions for obtaining optimal solutions from the algorithms are discussed in detail. The author describes computational complexity and some properties of the algorithms. The idea of the algorithms is presented with a few examples.

Keywords: *movement scheduling and synchronization, shortest paths, disjoint paths, multicriteria shortest paths problems, scheduling algorithms*