Zeszyty Naukowe WWSI, No 15, Vol. 10, 2016, s. 53-65

Symulowane wyżarzanie dla problemu harmonogramowania projektu z ograniczonymi zasobami

Marcin Klimek*

Państwowa Szkoła Wyższa w Białej Podlaskiej, Zakład Informatyki

Abstrakt

W artykule przedstawiony jest problem harmonogramowania projektu z ograniczonymi zasobami z kryterium minimalizacji czasu trwania przedsięwzięcia. Do rozwiązania zagadnienia stosowany jest algorytm symulowanego wyżarzania, którego skuteczność testowana jest przy wykorzystaniu standardowych zadań testowych. Eksperymenty przeprowadzane są przy różnych konfiguracjach algorytmu w celu ustalenia najlepszych parametrów: schematu chłodzenia, technik przeszukiwania (ruchów), schematów generowania rozwiązań.

Słowa kluczowe – symulowane wyżarzanie, harmonogramowanie projektu z ograniczonymi zasobami, procedury generowania rozwiązań

1. Wprowadzenie

Jednym z ważnych zagadnień praktycznych, często podejmowanych w badaniach, jest problem harmonogramowania projektu (przedsięwzięcia) [1]. Jest to ustalanie czasów rozpoczęcia lub zakończenia zadań projektowych (czynności), przydziału zasobów (maszyn, pracowników) do realizacji zadań przy przyjętych kryteriach optymalizacyjnych czasowych, finansowych. Na podstawie ustalonego uszeregowania

* E-mail: marcin_kli@interia.pl.

Zgłoszono do druku 16 października 2016 r.

tworzone są zamówienia materiałów niezbędnych do wykonywania zadań, przygotowywane są środki pieniężne na realizację czynności itp. Spośród analizowanych problemów badawczych jednym z najczęściej rozpatrywanych jest zagadnienie harmonogramowania projektu z ograniczoną dostępnością zasobów RCPSP (ang. *Resource-Constrained Project Scheduling Problem*) z kryterium minimalizacji czasu trwania wszystkich zadań (*makespan*).

W pracy rozważany jest klasyczny model optymalizacyjny RCPSP, który jako uogólnienie NP-trudnego problemu *Job Shop* jest także zagadnieniem NP-trudnym [2]. W związku ze złożonością problemu dla zagadnień z większą liczbą czynności uzasadnione jest użycie algorytmów heurystycznych, przybliżonych (przegląd i analiza algorytmów dla RCPSP w pracach [3-4]):

- konstrukcyjnych tj. algorytmy priorytetowe, wstawień [4] itp.,
- metaheurystycznych tj. algorytmy genetyczne [5], symulowanego wyżarzania [6-9], tabu search [10] i inne [11-13].

W pracy wykorzystany jest jeden z efektywnych algorytmów metaheurystycznych tj. algorytm symulowanego wyżarzania *SA* (ang. *Simulated Annealing*), który przetestowany jest dla standardowych zadań testowych z biblioteki *PSPLIB* (ang. *Project Scheduling Problem LIBrary*) [14].

2. Sformułowanie problemu

Rozważane jest klasyczne zagadnienie harmonogramowania projektu RCPSP, w którym przedsięwzięcia przedstawiane są jako acykliczny, skierowany graf $G(V, E)$ w reprezentacji „czynność na węźle" AON (ang. *Acivity On Node*), w którym zbiór V zawiera węzły reprezentujące czynności zbiór E zawiera łuków opisujące relacje kolejnościowe między czynnościami typu koniec-początek bez zwłoki (ang. *finish-start zero-lag precedence*). Projekt jest realizowany przy użyciu ograniczonej liczby odnawialnych zasobów, przy ustalonym jednym sposobie wykonania niepodzielnych zadań (ang. *single-mode, nonpreemptive*).

Przyjętą funkcją celu jest minimalizacja czasu trwania przedsięwzięcia (ang. *makespan minimisation*) równa czasowi rozpoczęcia czynności pozornej n+1 – wierzchołka końcowego w grafie $G(V, E)$. Minimalizacja F:

$$F = s_{n+1}. \tag{1}$$

Przyjmuje się następujące ograniczenia:

- ograniczenia zasobowe:

$$\sum_{i \in J(t)} r_{ik} \le a_k, \qquad \forall t = 1,...,s_{n+1}, \forall k = 1,...,K, \tag{2}$$

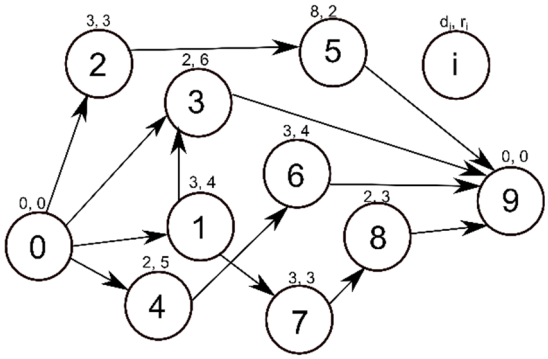
– ograniczenia kolejnościowe:

$$s_i + d_i \le s_j \qquad \forall (i, j) \in E \ , \tag{3}$$

gdzie:

n – liczba czynności,
i – numer (indeks) czynności, i = 1, …, n,
s_i – czas rozpoczęcia zadania i,
$J(t)$ – zbiór zadań wykonywanych w okresie [t-1, t] ,
K – liczba typów zasobów odnawialnych,
k – indeks (numer) typu zasobu, k = 1, …, K,
r_{ik} – zapotrzebowanie czynności i na zasób typu k,
a_k – dostępność zasobów odnawialnych typu k,
d_i – czas trwania czynności i.

Przykładowy projekt w reprezentacji AON, który posłuży do ilustracji rozpatrywanego zagadnienia, przedstawiony jest na rysunku 1. Projekt ten składa się z 8 zadań – węzły 0 i 9 reprezentują wierzchołek początkowy i końcowy grafu $G(V, E)$, nie są zadaniami projektowymi. Jest wykonywany przy użyciu jednego typu zasobów o dostępności równej 10.



Rysunek 1. Sieć AON dla przykładowego projektu

W reprezentacji AON jako węzły przedstawione są zadania z określonymi czasami ich trwania i zapotrzebowaniem na zasoby np. dla zadania 3 czas trwania jest równy 2, zapotrzebowanie na zasoby wynosi 6. Krawędzie reprezentują relacje kolejnościowe między zadaniami – np. krawędź od węzła 4 do 6 oznacza, że zadanie 6 może się rozpocząć po zakończeniu zadania 4.

3. Reprezentacja i procedury generowania harmonogramu

Rozwiązaniem problemu harmonogramowania projektu z ograniczonymi zasobami z kryterium minimalizacji czasu trwania prac jest wektor czasów rozpoczęcia lub zakończenia czynności tzw. reprezentacja bezpośrednia. W algorytmach lokalnych poszukiwań stosowane są reprezentacje pośrednie. Najlepsze rezultaty dla RCPSP [3], [4] są osiągane dla reprezentacji w postaci listy czynności (ang. *activity list*), w której rozwiązanie jest permutacją numerów kolejnych czynności przy uwzględnieniu zależności kolejnościowych. Rozwiązania w reprezentacji pośredniej muszą być przekształcane do reprezentacji bezpośredniej przy użyciu procedur dekodujących. Opracowywane są schematy generowania uszeregowania SGS (ang. *Schedule Generation Scheme*), które dekodują listę czynności w realizowalne harmonogramy (ustalane są wektory czasów rozpoczęcia zadań) przy uwzględnieniu ograniczeń czasowych i zasobowych.

Do procedur SGS zaliczyć można [15]:

- procedurę szeregową (ang. *serial SGS*), w której w kolejnych chwilach *t* ustalany jest czas rozpoczęcia dla pierwszego nieuszeregowanego zadania z listy czynności, w najwcześniejszym możliwym czasie przy uwzględnieniu ograniczeń kolejnościowych i zasobowych,
- procedurę równoległą (ang. *parallel SGS*), w której w kolejnych chwilach *t* rozpoczynane są wszystkie nieuszeregowane zadania (rozpatrywane w kolejności na liście czynności), które mogą być rozpoczęte w chwili *t* przy uwzględnieniu ograniczeń kolejnościowych i zasobowych.

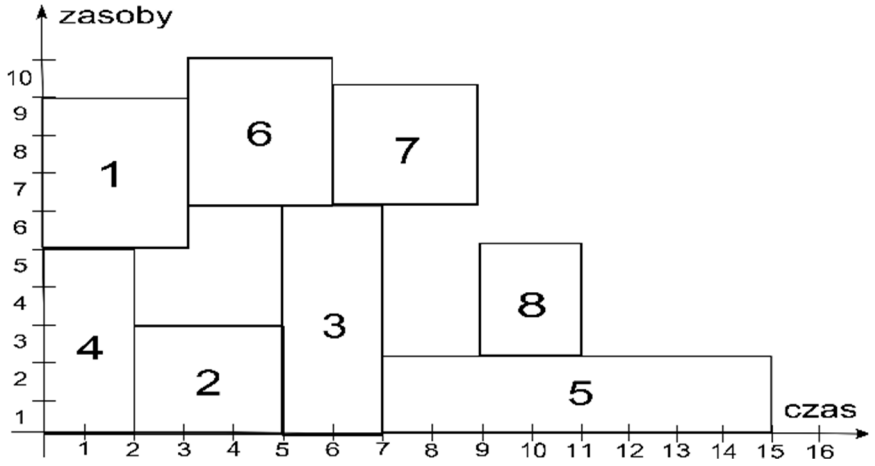
Przy dekodowaniu listy stosowane mogą być różne techniki harmonogramowania:

- w przód (ang. *forward scheduling*) – szeregowanie kolejnych czynności od początku listy czynności,
- wstecz (ang. *backward scheduling*) – szeregowanie kolejnych zadań od końca listy czynności, ustalając czasy rozpoczęcia zadań przy przyjętym terminie zakończenia projektu (*due date*).

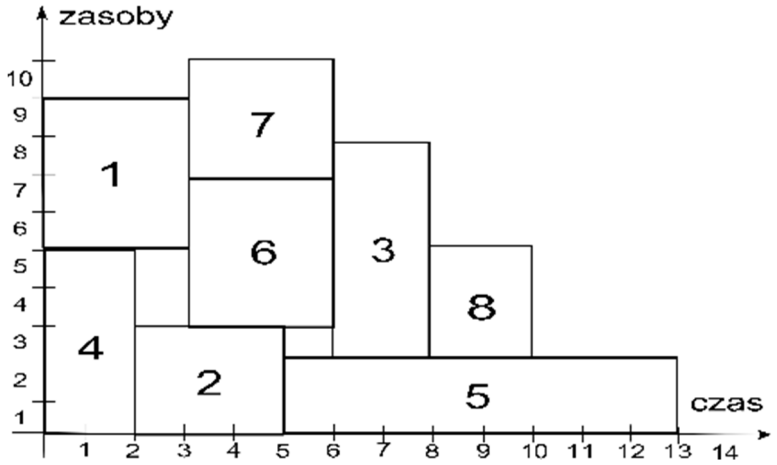
Dla danej listy czynności zastosowanie różnych procedur dekodujących często prowadzi do znalezienia harmonogramów o różnej jakości.

Niech lista czynności jest następująca: {1, 4, 2, 6, 3, 7, 5, 8}. Dla tej listy ustalany jest wektor czasów rozpoczęcia (harmonogram w reprezentacji bezpośredniej) przy wykorzystaniu SGS. Zadania są rozpatrywane od początku listy czynności {1, 4, 2, 6, 3, 7, 5, 8} (szeregowanie w przód) lub od jej końca {8, 5, 7, 3, 6, 2, 4, 1} (szeregowanie wsteczne). Przy harmonogramowaniu wstecznym przyjęto termin zakończenia projektu równy 20 – po wygenerowaniu harmonogramu czasy rozpoczęcia zadań są tak korygowane, aby czas rozpoczęcia projektu wynosił 0.

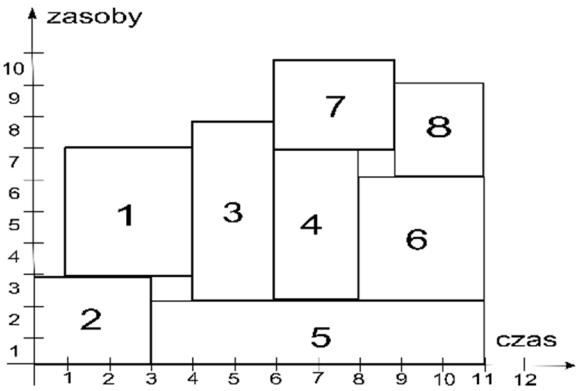
Uszeregowania znalezione dla listy czynności{1, 2, 3, 6, 4, 5, 7, 8} przy użyciu poszczególnych procedur znajdują się na rysunkach 2-5.



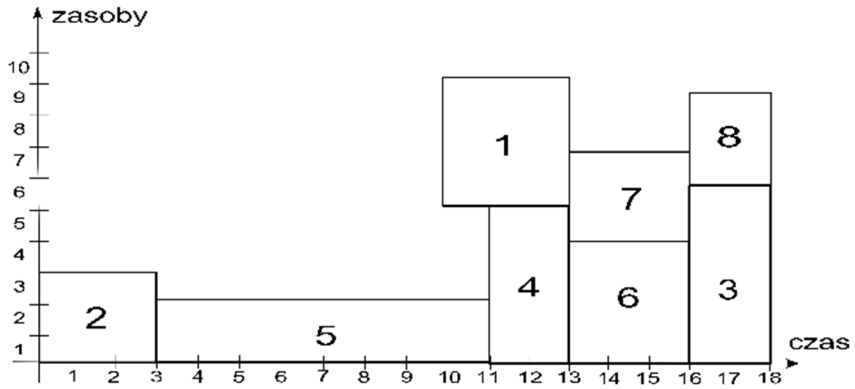
Rysunek 2. Harmonogram wygenerowany dla listy czynności {1, 4, 2, 6, 3, 7, 5, 8} – procedura szeregowa, harmonogramowanie w przód



Rysunek 3. Harmonogram wygenerowany dla listy czynności {1, 4, 2, 6, 3, 7, 5, 8} – procedura równoległa, harmonogramowanie w przód



Rysunek 4. Harmonogram wygenerowany dla listy czynności {1, 4, 2, 6, 3, 7, 5, 8} – procedura szeregowa, harmonogramowanie wsteczne



Rysunek 5. Harmonogram wygenerowany dla listy czynności {1, 4, 2, 6, 3, 7, 5, 8} – procedura równoległa, harmonogramowanie wsteczne

Zaobserwować można znaczne różnice w jakości harmonogramów – mierzonej czasem trwania wszystkich zadań. Najlepsze rozwiązanie (optymalne, z minimalnym czasem trwania projektu wynoszącym 11 – równym sumie czasów realizacji zadań ze ścieżki krytycznej 2 i 5) znalezione jest przy harmonogramowaniu wstecznym i szeregowej procedurze SGS. Wybór SGS ma wpływ na jakość wygenerowanych uszeregowań i wskazane jest przetestowanie różnych procedur SGS dla RCPSP.

4. Symulowane wyżarzanie

Inspiracją dla algorytmu symulowanego wyżarzania *SA* [16] jest proces wyżarzania ciał stałych. W trakcie przebiegu algorytm zmierza w kierunku minimalnego stanu energii. Charakteryzuje go możliwość „opuszczania" ekstremów lokalnych – stosowana jest funkcja akceptacji dopuszczająca przyjmowanie gorszych rozwiązań, zwłaszcza w początkowej fazie działania *SA*.

Algorytm *SA* jest heurystyką, nie daje gwarancji znalezienia rozwiązania optymalnego, ale znajduje bardzo dobre rozwiązania w akceptowalnym czasie, co potwierdzają liczne badania, także dla problemu RCPSP [3-4, 6-9]. W pracach badawczych pojawia się wiele wersji algorytmu *SA*. W tej pracy zaimplementowany jest algorytm zbliżony do podstawowej wersji [16] (patrz: rysunek 6).

```
Ustaw x, T0, λ, Tk;                                  //inicjalizacja, opcjonalnie faza strojenia
x* := x;
T := T0;
REPEAT
        Wybierz sąsiednie rozwiązanie y dla x;                    //wykonaj ruch
        IF( f(y) < f(x*) ) then x* := y;                          //zapamiętaj najlepsze
        IF( P(f(y), f(x), T) > rand ) THEN x := y;                //zaakceptuj
        Zaktualizuj temperaturę bieżącą T zgodnie z przyjętym schematem chłodzenia;
UNTIL (T < Tk)
Zwróć harmonogram S*
```

Rysunek 6. Algorytm symulowanego wyżarzania. Przyjęte oznaczenia: T – temperatura bieżąca, T_0 – temperatura początkowa (maksymalna), T_k – temperatura końcowa (minimalna), *rand* – liczba losowa z przedziału (0, 1), x – rozwiązanie bieżące w postaci listy czynności, y – rozwiązanie sąsiednie dla x, $f(x)$ – wartość funkcji celu F (czas trwania projektu) dla listy czynności x, $P(f(y), f(x), T)$ – funkcja akceptacji dla nowego rozwiązania y przy bieżącym rozwiązaniu x, x^*– najlepsze rozwiązanie w postaci listy czynności.

W zaimplementowanym algorytmie w danej temperaturze sprawdzane jest jedno rozwiązanie *y,* które jest zapamiętywane jako x^*, jeśli jest najlepszym dotychczas znalezionym rozwiązaniem. Rozwiązanie y jest zapisywane jako rozwiązanie

bieżące x w przypadku spełnienia kryterium akceptacji. Stosowana jest klasyczna funkcja akceptacji:

$$P(f(y), f(x), T) = \exp(\frac{f(x) - f(y)}{T}). \tag{4}$$

Przy tak zdefiniowanej funkcji akceptacji (patrz: wzór 4) rozwiązanie y o niższej lub równej wartości funkcji celu F ($f(x) \geq f(y)$) jest zawsze akceptowane, ponieważ zachodzą wtedy nierówności: $P(f(y), f(x), T) \geq 1 > rand$. Rozwiązanie y gorsze od x jest akceptowane, w przypadku gdy wygenerowana liczba losowa *rand* jest mniejsza od prawdopodobieństwa akceptacji $P(f(y), f(x), T)$ – większego na początku działania *SA* przy wyższej temperaturze bieżącej T.

W algorytmie *SA,* podobnie jak w innych metaheurystykach, ważną rolę spełnia dobór technik przeszukiwania przestrzeni rozwiązań, generowania rozwiązań sąsiednich. Dla reprezentacji w postaci listy czynności, opracowywane są liczne operatory przeszukiwania (ruchy), zbliżone do znanych dla reprezentacji permutacyjnej, ale uwzględniające zależności kolejnościowe występujące między zadaniami. W tej pracy stosowane są ruchy:

- **zamień** – wybranie losowego zadania i, które jest zamieniane z losowo wybranym innym zadaniem znajdującym się na liście czynności na pozycji między ostatnim poprzednikiem zadania i a pierwszym następnikiem i,
- **zamień sąsiednie** – wybranie losowego zadania, które jest zamieniane z zadaniem kolejnym na liście czynności, jeśli operacja ta nie narusza ograniczeń kolejnościowych,
- **wstaw** – wybranie losowego zadania, które jest przesuwane na losową pozycję między ostatnim na liście czynności jego poprzednikiem a pierwszym jego następnikiem.

W zaimplementowanym algorytmie warunkiem zatrzymania jest osiągnięcie temperatury bieżącej T równej temperaturze końcowej T_k. Zmiana temperatury bieżącej T odbywa się zgodnie z przyjętym schematem chłodzenia:

- liniowym:

$$T_{p+1} = T_p - \lambda, \quad \text{dla} \quad \lambda = \frac{T_k - T_0}{N} \tag{5}$$

- geometrycznym:

$$T_{p+1} = \lambda \cdot T_p, \quad \text{dla} \quad \lambda = (\frac{T_k}{T_0})^{\frac{1}{N}} \tag{6}$$

– logarytmicznym:

$$T_{p+1} = \frac{T_p}{1 + \lambda \cdot T_p}, \qquad \text{dla} \quad \lambda = \frac{T_0 - T_k}{N \cdot T_0 \cdot T_k} \tag{7}$$

gdzie:

T_p – temperatura bieżąca T w p-tym przebiegu *SA*,
p – numer przebiegu,
N – maksymalna liczba przebiegów,
λ – parametr lambda schematu chłodzenia.

W analizowanej wersji *SA* liczba przeglądanych rozwiązań jest równa liczbie przebiegów N.

5. Wyniki eksperymentów

Eksperymenty przeprowadzano przy użyciu aplikacji zaimplementowanej w języku C# w środowisku Visual Studio.NET dla 480 instancji testowych ze zbioru J30 (problemy 30-zadaniowe) oraz dla 480 instancji ze zbioru J90 (problemy 90-zadaniowe) z biblioteki *PSPLIB* na komputerze z procesorem Intel Core I7 CPU 3.4 GHz, 16GB RAM.

Celem eksperymentów jest ustalenie skutecznych parametrów algorytmu symulowanego wyżarzania dla problemu minimalizacji czasu trwania projektu. Dobór dotyczy parametrów schematu chłodzenia, technik przeszukiwania przestrzeni rozwiązań (ruchy), przy czym szukano najlepszych ustawień algorytmu *SA* dla poszczególnych strategii harmonogramowania: w przód i wstecz, przy szeregowej i równoległej procedurze SGS oraz dla strategii mieszanych.

W fazie strojenia *SA* tworzonych jest 100 losowych uszeregowań, z których najlepsze jest rozwiązaniem startowym. Dla każdego z eksperymentów liczba przeglądanych rozwiązań wynosi 5000 (podobnie jak w innych badaniach dla tego zagadnienia [3-4]). Testowane są różne temperatury początkowe: 5, 1, 0.5 lub 0.1, przy schematach chłodzenia: logarytmicznym, geometrycznym lub liniowym. Do generowania nowych rozwiązań stosowane są ruchy: zamień, wstaw lub zamień sąsiednie. Wyniki obliczeń dla najlepszych konfiguracji *SA,* przy różnych strategiach harmonogramowania, umieszczone są w tabelach 1 i 2.

Tabela 1. Wyniki eksperymentów obliczeniowych dla projektów ze zbioru J30. Oznaczenia kolumn: *a* – odchylenie średniej wartości funkcji celu *F* od rozwiązania optymalnego, *b* – liczba rozwiązań optymalnych (spośród 480 eksperymentów), *c* – średni czas trwania obliczeń (w sekundach).

Procedura SGS	***SA*(schemat chłodzenia, ruch, temperatura początkowa)**	***a***	***b***	***c***
Szeregowa w przód	logarytmiczny, zamień, *T*o=0.1	0.41%	406	0.048
Równoległa w przód	geometryczny, zamień, *T*o=0.5	1.28%	308	0.193
Szeregowa wstecz	geometryczny, zamień, *T*o=1	**0.34%**	**422**	**0.069**
Równoległy wstecz	logarytmiczny, zamień, *T*o=0.1	2.39%	188	0.263
W przód (szer. i równ.)	geometryczny, zamień, *T*o=0.5	0.45%	409	0.128
Wstecz (szer. i równ.)	liniowy, zamień, *T*o=0.5	0.47%	405	0.183
Mieszane	liniowy, zamień, *T*o=1	0.49%	403	0.177

Tabela 2. Wyniki eksperymentów obliczeniowych dla projektów ze zbioru J90. Oznaczenia kolumn: *a* – odchylenie średniej wartości funkcji celu *F* od najlepszych znanych rozwiązań, *b* – liczba rozwiązań identycznych z najlepszymi znanymi (spośród 480 eksperymentów), c – średni czas trwania obliczeń (w sekundach).

Procedura SGS	***SA*(schemat chłodzenia, ruch, temperatura początkowa)**	***a***	***b***	***c***
Szeregowa w przód	liniowy, wstaw, *T*o=1	2.53%	342	0.205
Równoległa w przód	logarytmiczny, wstaw, *T*o=0.1	2.80%	276	1.830
Szeregowa wstecz	logarytmiczny, wstaw, *T*o=1	**2.21%**	**344**	**0.298**
Równoległy wstecz	geometryczny, wstaw, *T*o=0.5	3.41%	189	1.957
W przód (szer. i równ.)	geometryczny, wstaw, *T*o=5	2.66%	321	1.063
Wstecz (szer. i równ.)	liniowy, wstaw, *T*o=0.1	2.66%	335	1.204
Mieszane	liniowy, wstaw, *T*o=5	2.65%	338	1.205

Eksperymenty wykazały, że największy wpływ na wartość funkcji celu *F* (czasu trwania projektu) ma dobór procedury dekodującej. Lepszej jakości uszeregowania tworzone są przy użyciu mniej czasochłonnej szeregowej procedury SGS, zarówno przy harmonogramowaniu w przód jak i wstecz. Zastosowanie strategii mieszanych

(na przemian generowanie rozwiązań przy użyciu różnych procedur SGS) nie prowadzi do znalezienia lepszych uszeregowań niż wygenerowanych przez procedury szeregowe.

Dla projektów ze zbioru J30 najlepsze rozwiązania są znalezione przy użyciu harmonogramowania wstecznego z szeregową procedurą dekodującą z geometrycznym schematem chłodzenia, ruchem zamień i temperaturą początkową T_o=1. Rozwiązania te są jedynie o 0,34% gorsze od harmonogramów optymalnych (są to wyniki zbliżone do najlepszych metaheurystyk [3-4]), przy czym najlepsze uszeregowania wygenerowane są dla 422 instancji testowych spośród 480 analizowanych.

Dla projektów ze zbioru J90 najlepsze uszeregowania są generowane przy użyciu harmonogramowania wstecznego z szeregową procedurą dekodującą z logarytmicznym schematem chłodzenia, ruchem wstaw i temperaturą początkową T_o=1. Rozwiązania te są jedynie 2,21% gorsze od najlepszych znanych harmonogramów.

Dla różnych strategii harmonogramowania i procedur SGS różne konfiguracje algorytmu *SA* są najkorzystniejsze (różne schematy chłodzenia i ustawienia temperatury początkowej). Dla projektów ze zbioru J30 najlepsze rezultaty osiągane są dla ruchu zamień, przy projektach ze zbioru J90 dla ruchu wstaw.

6. Wnioski

W artykule przedstawiony jest algorytm symulowanego wyżarzania dla problemu harmonogramowania projektu z minimalizacją czasu trwania wszystkich zadań. Algorytm ten testowany jest przy różnych ustawieniach schematu chłodzenia, ruchów, procedur dekodujących. Eksperymenty wykazały, że największy wpływ na jakość rozwiązań ma dobór procedury SGS, najlepszy wyniki są osiągane przy szeregowej procedurze dekodującej.

Podjęte zagadnienie jest często podejmowane i istotne z praktycznego punktu widzenia.

Bibliografia

[1] Hartmann S., Briskorn D., *A Survey of Variants and Extensions of the Resource-Constrained Project Scheduling Problem*, "European Journal of Operational Research" Vol. 207, No. 1, 2012

[2] Błażewicz J., Lenstra J., Kan A.R., *Scheduling subject to resource constraints – classification and complexity*, "Discrete Applied Mathematics" Vol. 5, 1983

[3] Hartmann S., Kolisch R., *Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem*, "European Journal of Operational Research" Vol. 127, 2000

[4] Kolisch R., Padman R., *An integrated survey of deterministic project scheduling*, "OMEGA The International Journal of Management Science" Vol. 29, 2001

[5] Klimek M., *A genetic algorithm for the project scheduling with the resource contraints*, "Annales UMCS Informatica" Vol. 10, nr 1, 2010

[6] Klimek M., *Predyktywno-reaktywne harmonogramowanie produkcji z ograniczoną dostępnością zasobów*, praca doktorska, AGH Kraków, 2010

[7] Boctor F.F., *Resource-constrained project scheduling by simulated annealing*, "International Journal of Operational Research" Vol. 34, No. 8, 1996

[8] Bouleimen K., Lecocq H., *A new efficient simulated annealing algorithm for the resource constrained project scheduling problem and its multiple version*, "European Journal of Operational Research" Vol. 149, 2003

[9] Mika M., Waligóra G., Węglarz J., *Simulated annealing and tabu search for multi-mode resource-constrained Project scheduling with positive discounted cash flows and different payment models*, "European Journal of Operational Research" Vol. 164, No. 3, 2005

[10] Thomas P. R., Salhi S., *A Tabu Search Approach for the Resource Constrained Project Scheduling Problem*, "Journal of Heuristics" Vol. 4, 1998

[11] Akbari R., Zeighami V., Ziarati K. *Artificial bee colony for resource constrained project scheduling problem*, "International Journal of Industrial Engineering Computations", Vol. 2, No. 1, 2011

[12] Eshraghi A., *A new approach for solving resource constrained project scheduling problems using differential evolution algorithm*, "International Journal of Industrial Engineering Computation s" Vol. 7, 2016

[13] Koulinas, G., Kotsikas, L., Anagnostopoulos, K., *A particle swarm optimization based hyperheuristic algorithm for the classic resource constrained project scheduling problem*, "Information Sciences" Vol. 277, 2014

[14] Kolisch R., Sprecher A.: *PSPLIB – a project scheduling library*, "European Journal of Operational Research" Vol. 96, 1997

[15] Kolisch R., *Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation*, "European Journal of Operational Research" Vol. 90, 1996

[16] Kirkpatrick S., Gelatt C.D., Vecchi M.P., *Optimization by simulated annealing*, "Science" Vol. 220, 1983

Simulated annealing for project scheduling with limited resources

Abstract

In this paper resource-constrained project scheduling problem with optimisation criterion of minimising makespan is presented. To solve the problem is applied simulated annealing algorithm, whose effectiveness is tested using standard test instances. Experiments are performed with different configurations algorithm to determine the best parameters: cooling schemes, search techniques (moves), schedule generation schemes.

Keywords – simulated annealing, resource-constrained project scheduling, schedule generation schemes