ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2008
Seria: AUTOMATYKA z. 150 Nr kol. 1796

Marcin KLIMEK
Państwowa Wyższa Szkoła Zawodowa, Biała Podlaska
Piotr ŁEBKOWSKI
Akademia Górniczo-Hutnicza

ALGORYTMY METAHEURYSTYCZNE DLA PROBLEMU HARMONOGRAMOWANIA PROJEKTU Z KAMIENIAMI MILOWYMI

Streszczenie. W artykule zaproponowano model matematyczny problemu harmonogramowania projektu z ograniczoną dostępnością zasobów RCPSP (ang. *Resource-Constrained Project Scheduling Problem*), który uwzględnia system kamieni milowych. Dla części zadań (związanych z kamieniami milowymi) określono nieprzekraczalne terminy ich zakończenia. Opracowano funkcję celu, która uwzględnia terminy realizacji wszystkich tych czynności. Dla zdefiniowanego problemu przetestowano skuteczność działania algorytmów genetycznych i symulowanego wyżarzania.

METAHEURISTIC ALGORITHMS FOR PROJECT SCHEDULING PROBLEM WITH MILESTONES

Summary. In article is proposed mathematical model for Resource-Constrained Scheduling Problem with milestones. For some activities (related to the milestones), unsurpassable term of completion is determined. We have defined objective function, taking into consideration the observance of the times of completion of all these activities. For defined problem we have tested effectiveness of genetic and simulated annealing algorithms.

1. Wprowadzenie

W praktyce przemysłowej coraz częściej stosowana jest produkcja na zlecenie (ang. *MTO – Make-To-Order*). Tak wytwarzane są zwłaszcza wyroby niestandardowe, dla odbiorcy indywidualnego, którego wymagania są zmienne i bywają nieprzewidywalne. Każde zlecenie produkcyjne może być traktowane jako osobny projekt powstający w konsultacji z klientem. Realizacja takiego projektu obarczona jest niepewnością związaną z unikalnością realizowanych zadań. W związku z tym występują problemy z oszacowaniem terminu realizacji poszczególnych etapów prac czy też całego przedsięwzięcia. Klienci, w celu zmniejszenia ryzyka niepowodzenia całego przedsięwzięcia, często wymagają od wykonawców (zwłaszcza przy dużych przedsięwzięciach) określenia punktów kontroli przebiegu prac, tzw. kamieni milowych, dla których definiowane są nieprzekraczalne terminy ich wykonania.

Opóźnienia w realizacji zadań mogą pociągać za sobą kary umowne, natomiast dotrzymanie umownych terminów może się wiązać z zapłatą za wykonanie danego etapu projektu.

W niniejszym artykule zaproponowano matematyczny model problemu harmonogramowania projektu z ograniczoną dostępnością zasobów RCPSP ze zdefiniowanymi, nieprzekraczalnymi terminami realizacji części zadań związanych z kamieniami milowymi. Zaproponowany model może być bardzo użyteczny przy realizacji dużych zleceń produkcyjnych, konstrukcyjnych czy rozwojowych. W celu rozwiązania badanego problemu opracowano i przetestowano algorytmy metaheurystyczne, tj. algorytm genetyczny GA (ang. *Genetic Algorithm*) i algorytm symulowanego wyżarzania SA (ang. *Simulated Annealing*).

2. Sformułowanie problemu

Projekt to zbiór współzależnych zadań (operacji, czynności), realizowany przy użyciu ograniczonej liczby zasobów. Zasoby są odnawialne [8] (ang. *renewable*), tzn. ilość zasobu jest stała niezależnie od obciążeń w poprzednich okresach. W każdym momencie czasu t wykorzystanie zasobów przez czynności nie przekracza wielkości dostępnych. To ograniczenie wyraża wzór (1) [2]:

$$\sum_{i \in S_t} r_{ik} \le a_k \qquad \forall t, \forall k \tag{1}$$

gdzie:
a_k – liczba dostępnych zasobów typu k,
S_t – zbiór zadań wykonywanych w przedziale czasu $[t\text{-}1, t]$,
r_{ik} – zapotrzebowanie czynności i na zasób typu k.

Dla rozważanego problemu RCPSP stosowana będzie sieć *AON*, tzw. sieć czynności, która jest wykorzystywana dla problemów szeregowania z kryterium optymalizacji czasu. Projekty w sieci *AON* reprezentowane są jako acykliczny, spójny, prosty graf skierowany $G(V, E)$, w którym V oznacza zbiór węzłów odpowiadający czynnościom, a E to zbiór łuków opisujących zależności kolejnościowe. Czynności ponumerowane są od 1 do n, w taki sposób, że poprzednik ma zawsze niższy numer od następnika (porządek topologiczny). Do grafu G dodawane są dwa fikcyjne zadania 0 i n+1, reprezentujące odpowiednio początek i koniec projektu. Zadania są niepodzielne (ang. *nonpreemptive*) i istnieje tylko jeden sposób ich wykonania (ang. *single-mode RCPSP*).

Między zadaniami występują relacje typu koniec-początek bez zwłoki [2] (ang. *finish-start, zero-lag precedence*) – następnik może rozpocząć się bezzwłocznie po zakończeniu operacji poprzedniej. Ograniczenia kolejnościowe można opisać wzorem (2):

$$s_i + d_i \le s_j \qquad \forall (i, j) \in E \tag{2}$$

gdzie:
s_i – czas rozpoczęcia zadania i,
d_i – czas wykonywania zadania i.

Problem harmonogramowania projektu z ograniczoną dostępnością zasobów RCPSP polega na znalezieniu wektora terminów rozpoczęcia (lub zakończenia) dla przyjętego kryterium optymalizacyjnego. Dla problemu RCPSP stosowane są procedury dekodujące SGS (ang. SGS – *Schedule Generation Scheme*), które zamieniają listę czynności (lub listę priorytetową) w harmonogram, uwzględniając ograniczenia kolejnościowe i zasobowe. Najczęściej wykorzystywane są dwa schematy generowania harmonogramu [5]: szeregowy (ang. *serial SGS*) oraz równoległy (ang. *parallel SGS*).

Kryterium optymalizacyjnym w badaniach dotyczących RCPSP najczęściej jest minimalizacja łącznego czas trwania całego projektu (ang. *makespan*). Podejmowane jest również zagadnienie terminowej realizacji całego przedsięwzięcia. W niniejszym artykule proponowane jest podejście, w którym celem jest terminowa realizacja umownych punktów kontroli przebiegu prac tzw. kamieni milowych. Zastosowanie systemu kamieni milowych w problemie RCPSP można sprowadzić do określenia dla części zadań, związanych z kamieniami milowymi, nieprzekraczalnych terminów ich zakończenia (w szczególności terminu zakończenia całego projektu δ_{n+1}) [4]:

$$z_i \le \delta_i \tag{3}$$

gdzie:
z_i – czas zakończenia czynności i,
δ_i – nieprzekraczalny termin zakończenia czynności i.

Z praktycznego punktu widzenia, w związku z niepewnością występującą przy realizacji projektu, użyteczne jest znalezienie takiego harmonogramu zadań, aby maksymalnie zabezpieczyć terminową realizację kamieni milowych [4]. W celu zdefiniowania funkcji celu realizującej to założenie oznaczmy przez pb_i poziom bezpieczeństwa dla realizacji kamienia milowego km_i wyznaczany następująco:

$$pb_i = \frac{rez_i}{tkm_i} \tag{4}$$

gdzie:
rez_i – różnica między nieprzekraczalnym terminem zakończenia δ_j (określonym dla km_i) a najwcześniejszym możliwym terminem wykonania wszystkich czynności, których wykonanie jest niezbędne do realizacji danego kamienia milowego km_i.

Proponowaną funkcją celu F, uwzględniającą zabezpieczenie terminowego wykonania wszystkich etapów projektu, jest ważona suma poziomu zabezpieczenia kamieni milowych pb_i określona wzorem (5):

$$F = \{\sum_{i=1}^{m} pb_i \cdot wm_i\} \tag{5}$$

gdzie:
wm_i – waga przypisana kamieniowi milowemu km_i.

Wartość wagi wm_i zależy od aktualnego poziomu zabezpieczenia kamienia milowego km_i i ustalana jest na podstawie posortowanej rosnąco według pb_i listy kamieni milowych, którym kolejno przypisuje się malejące wm_i. W pracy przyjęto

jako wm_i kolejne liczby naturalne od m do 1. Kamieniom milowym można przypisać inne wagi, przy czym powinien być spełniony warunek: większa waga wm_i dla mniej zabezpieczonych km_i.

Problem harmonogramowania projektu z ograniczoną dostępnością zasobów RCPSP, jako uogólnienie klasycznego problemu *job shop*, jest zadaniem silnie NP-trudnym. Dla projektów z większą liczbą czynności stosowane są algorytmy przybliżone, w szczególności algorytmy metaheurystyczne, tj. algorytmy genetyczne, algorytmy symulowanego wyżarzania. Metaheurystyki nie wnikają w strukturę problemu, próbkując obiecujące rozwiązania z przestrzeni wszystkich dozwolonych. Algorytmy metaheurystyczne skuteczne dla problemu RCPSP z kryterium minimalizacji *makespan* powinny być również skuteczne dla proponowanego problemu RCPSP z kamieniami milowymi z funkcją celu F określoną wzorem (5).

Przy rozwiązywaniu problemów optymalizacyjnym istotne jest znalezienie odpowiedniej reprezentacji potencjalnych rozwiązań (odpowiedniego kodowania). Dla rozważanego problemu stosowana będzie, dająca najlepsze wyniki w badaniach [7], reprezentacja permutacyjna, w której rozwiązaniem jest permutacja bez powtórzeń numerów zadań, tzw. lista czynności.

W dalszej części artykułu opisano testowane metaheurystyki: GA i SA. W celu oceny ich skuteczności ich działanie zostanie porównane do czysto losowej procedury generowania listy czynności (zwanej dalej procedurą *LosoweRCPSP*):

Krok 1: Ustawienie pustej listy zadań P, natomiast w liście L umieszczenie wszystkich następników czynności fikcyjnej 0.

Krok 2: Losowe wybranie zadania *i* z listy L na kolejne miejsce w liście P.

Krok 3: Usunięcie wylosowanego zadania *i* z listy L i dodanie do tej listy wszystkich następników zadania *i*.

Krok 4: Powtórzenie kroków 2-3 aż do wypełnienia listy P wszystkimi czynnościami niefikcyjnymi

Oznaczenia:

P – lista czynności poddawana następnie procedurze SGS,

L – lista dostępnych zadań niedodanych do listy P, takich które mogą być rozpoczęte (poprzednikami tego zadania są jedynie czynności już umieszczone na liście L).

Procedura *LosoweRCPSP* stosowana jest również przy generowaniu populacji początkowej dla GA i przy znajdowaniu rozwiązania początkowego dla SA.

3. Algorytm genetyczny

Idea algorytmu genetycznego została zaczerpnięta z nauk przyrodniczych. Sposób przeszukiwania potencjalnych rozwiązań naśladuje procesy naturalne: dziedziczenie genetyczne i zjawiska doboru naturalnego. Mechanizmy doboru naturalnego (selekcji) prowadzą do przetrwania osobników najlepiej przystosowanych w danym środowisku. Osobniki, które przetrwają – przekazują informację genetyczną swoim potomkom (operacje krzyżowania, mutacji). Kolejne pokolenia są przeciętnie coraz lepiej dostosowane do warunków środowiska. Schemat działania zaimplementowanego algorytmu genetycznego można przedstawić następująco:

Krok 1: Inicjalizacja.
Generowanie populacji początkowej (przy użyciu procedury *LosoweRCPSP*). Ustalenie parametrów algorytmu: prawdopodobieństwa krzyżowania, mutacji, liczby pokoleń, rozmiaru populacji i liczby osobników elitarnych.
Krok 2: Selekcja.
Wybór osobników do nowej populacji. Stosowane metody: rankingowa, koła ruletki, turniejowa, wszystkie ze strategią elitarną.
Krok 3: Krzyżowanie.
Wybór osobników (rodziców) do rozmnażania i następnie przeprowadzenie wymiany materiału genetycznego między losowo dobranymi rodzicami przy zastosowaniu wybranego operatora krzyżowania (PPX, 1PX lub 2PX przedstawione w rozdziale 6).
Krok 4: Mutacja.
Na wybranych osobnikach zmiany w liście zadań przy użyciu wybranego operatora mutacji (mutacje typu Wstaw, Zamień, Hartmanna lub Inwersja opisane w rozdziale 6).
Kroki 2-4 powtarzane są aż do spełnienia warunku STOP-u, np. przebieg przez określoną liczbę pokoleń.

W zagadnieniu optymalizacji genetycznej bardzo istotne jest właściwe określenie funkcji przystosowania. Jest to odpowiednio przekształcona funkcja celu F rozważanego problemu. W celu przeskalowania F, jako funkcję przystosowania przyjęto $fitness_i$ określone wzorem (6):

$$fitness_i = F_i - \min_{j=1...rozmiar\_populacji}(fitness_j) \tag{6}$$

gdzie:
F_i – wartość funkcji celu F określonej wzorem (5) dla osobnika i,
$fitness_i$ – wartość funkcji przystosowania osobnika i.

Funkcja przystosowania jest wykorzystywana do oceny przy wyborze osobników do nowej populacji. W pracy testowano następujące metody selekcji:

- Selekcja turniejowa – z całej populacji losowanych (ze zwracaniem) jest kilka chromosomów, najlepszy z nich wybierany jest do nowej populacji.
- Selekcja metodą koła ruletki, proporcjonalna – losowanie chromosomów z rozkładem opartym na funkcjach przystosowania.
- Selekcja rankingowa – tworzony jest ranking osobników zgodnie z rosnącą wartością funkcji przystosowania. Na podstawie rankingu każdemu osobnikowi przydzielana jest liczba tzw. szans. Najsłabszy osobnik otrzymuje jedną szansę, natomiast najsilniejszy liczbę szans równą rozmiarowi populacji. Do nowej populacji losowane są chromosomy zgodnie z rozkładem opartym na liczbie szans poszczególnych osobników.

4. Algorytm symulowanego wyżarzania

Idea działania algorytmów symulowanego wyżarzania pochodzi z termodynamiki i nawiązuje do procesu wyżarzania ciał stałych. Zastosowanie SA umożliwia unikanie ekstremów lokalnych przez możliwość akceptacji gorszych

rozwiązań od dotychczas znalezionych. Schemat działania zaimplementowanego algorytmu symulowanego wyżarzania można zaprezentować następująco [3]:
Krok 1: Inicjalizacja.
Generowanie początkowej listy (permutacji) zadań. Ustalenie w fazie strojenia parametrów algorytmu: temperatury początkowej, końcowej (minimalnej) oraz parametru lambdy.
Krok 2: Wybór z sąsiedztwa bieżącej listy zadań rozwiązania sąsiedniego: stosowane są tutaj techniki jednoargumentowe opisane w rozdziale 6.
Krok 3: Akceptacja z prawdopodobieństwem p_a nowego rozwiązania. Jeśli ma być zaakceptowane, to staje się ono rozwiązaniem bieżącym w następnej iteracji algorytmu. Zastosowano klasyczną funkcję akceptacji [3].
Krok 4: Zmiana temperatury zgodnie z przyjętym schematem chłodzenia (na początku temperatura jest równa temperaturze początkowej). Założono logarytmiczny schemat chłodzenia.

Kroki 2-4 powtarzane są aż do spełnienia warunku STOP-u. Warunkiem stopu może być przebieg zadanej liczby iteracji algorytmu lub osiągnięcie temperatury T równej temperaturze minimalnej T_k.

Kluczowe dla SA jest odpowiednie określenie parametrów algorytmu i znalezienie skutecznej techniki przeszukiwań potencjalnych rozwiązań (krok 2).

5. Techniki budowy nowych rozwiązań

Dla reprezentacji permutacyjnej powszechnie znane są operatory przeszukiwania przestrzeni rozwiązań. Dla różnych algorytmów metaheurystycznych stosowane są podobne techniki budowy nowych rozwiązań (zmiany aktualnych rozwiązań). W GA zwane są one operatorami genetycznymi, natomiast w algorytmach symulowanego wyżarzania ruchem. Algorytmy genetyczne stosują techniki wieloargumentowe (operujące na więcej niż jednym rozwiązaniu) i jednoargumentowe (operujące na jednym rozwiązaniu). GA operują bowiem na populacji osobników (rozwiązań). Algorytmy SA wykorzystują jedynie techniki jednoargumentowe.

Operatory genetyczne dla GA oraz ruchy w SA nie mogą zaburzać zależności kolejnościowych między zadaniami. Procedury SGS zawsze wygenerują poprawny harmonogram, ale przetwarzanie list zadań niespełniających ograniczeń kolejności jest nadmiarowe. Na przykład, jeśli czynność j jest następnikiem czynności i, procedury SGS dla list $P = \{..., j, ..., i, ...\}$ nie szeregują czynności j, aż do momentu kiedy będzie uszeregowana czynność i, czyli wygenerują identyczne harmonogramy dla listy P jak dla listy $P' = \{..., ..., i, j, ...\}$. Poniżej zaprezentowano stosowane techniki budowy nowych rozwiązań z podziałem na wielo- i jednoargumentowe.

5.1. Techniki wieloargumentowe

Jako operatory krzyżowania w GA zastosowano następujące techniki wieloargumentowe (wykorzystywane w badaniach również dla innych problemów optymalizacyjnych) [8]:

- 1PX (krzyżowanie jednopunktowe, ang. *One-Point Crossover*) – osobnik potomny powstaje przez skopiowanie genów od pierwszej pozycji do punktu krzyżowania od jednego z rodziców, a brakujące geny są uzupełniane w kolejności ich występowania u drugiego z rodziców.
- 2PX (krzyżowanie dwupunktowe, ang. *Two-Points Crossover*) – osobnik potomny powstaje przez skopiowanie genów od jednego z rodziców (poza fragmentem chromosomu między dwoma wylosowanymi punktami krzyżowania) i uzupełnienie brakujących genów w kolejności ich występowania u drugiego z rodziców.
- PPX [1] (krzyżowanie z zachowaniem następstwa, ang. *Precedence Preservance Crossover*) – na początku wyznaczany jest wektor V o długości równej rozmiarowi chromosomu, złożony z losowo wygenerowanych 0 i 1. Osobnik potomny powstaje przez kopiowanie kolejnych genów z chromosomów rodzicielskich R1 i R2 na podstawie wszystkich kolejnych wartości wektora V: gdy na danej pozycji w wektorze V jest wartość 0, do potomka gen jest kopiowany z R1, gdy jest wartość 1, gen z R2. Po dopisaniu do potomka dany gen jest usuwany z obu rodziców.

Dokonano wyboru takich operatorów krzyżowania, gdyż ich wykorzystanie nie zaburza zależności kolejnościowych [8].

5.2. Techniki jednoargumentowe

Techniki jednoargumentowe są stosowane zarówno w algorytmach genetycznych, jak i w algorytmach symulowanego wyżarzania.

W GA jako operatory mutacji zastosowano [8]:

- Mutację typu Wstaw (ang. *Insert*) – wybierana jest losowo czynność, która następnie jest przesuwana na losową pozycję między ostatnim w uporządkowaniu zadań poprzednikiem a pierwszym następnikiem.
- Mutację typu Zamień (ang. *Swap*) – wybierana jest losowo czynność, która następnie jest zamieniana z losowo wybranym zadaniem zajmującym pozycję między ostatnim w uporządkowaniu zadań poprzednikiem a pierwszym następnikiem.
- Mutację Hartmanna – zamiana przyległych zadań na sąsiednich pozycjach w permutacji zadań, wybierana jest losowo czynność, która następnie jest zamieniana z czynnością kolejną na liście, o ile nie naruszy to ograniczeń kolejnościowych.
- Mutację typu Inwersja (ang. *Inversion*) – odwrócenie losowo wybranego fragmentu listy czynności w taki sposób, aby nie naruszyć ograniczeń kolejnościowych.

Dla algorytmów symulowanego wyżarzania do przeszukiwania przestrzeni rozwiązań użyto wszystkich technik jednoargumentowych wykorzystanych w GA jako operatory mutacji. Dodatkowo zaimplementowano ruchy typu:

- Wstaw Wszystkie – wybierana jest losowo czynność, która następnie jest przesuwana na pozycję między ostatnim w uporządkowaniu zadań poprzednikiem a pierwszym następnikiem, sprawdzane są wszystkie możliwe pozycje wstawienia i wybierana jest pozycja, dla której funkcja kryterium osiąga największą wartość.

– Zamień Wszystkie – wybierana jest losowo czynność, która następnie jest zamieniana z każdym zadaniem zajmującym pozycję między ostatnim w uporządkowaniu zadań poprzednikiem a pierwszym następnikiem i wykonywana jest zamiana, dla której funkcja kryterium osiąga największą wartość.

6. Wyniki badań eksperymentalnych

Badania przeprowadzono przy użyciu 120 instancji testowych zaproponowanych w pracy [6] (po 60 dla problemów złożonych z 30 i 120 zadań). Dla każdej problemu testowego zdefiniowano dodatkowo cztery kamienie milowe, określając nieprzekraczalne terminy realizacji części zadań. Eksperymenty prowadzono na komputerze klasy Pentium z procesorem 1,7 GHz przy użyciu zaimplementowanego programu w języku C# w środowisku Visual Studio .NET.

Ze względu na stochastyczny charakter testowanych algorytmów dla każdej instancji testowej każdy z algorytmów był trzykrotnie uruchamiany. Aby móc porównać działanie testowanych algorytmów, w każdym eksperymencie przyjęto stałą liczbę wygenerowanych rozwiązań w procesie przeszukiwania przestrzeni rozwiązań równą 20000: 20000 iteracji w algorytmie SA, 1000 pokoleń przy rozmiarze populacji wynoszącej 20 osobników w algorytmie GA, 20000 losowo wygenerowanych list czynności przy użyciu procedury *LosoweRCPSP*.

Eksperymenty wykazały lepszą skuteczność szeregowej procedury dekodującej SGS niż równoległej procedury dekodującej (o 4,1% mniejsze średnie procentowe odchylenie względne od najlepszego uszeregowania zadania dla problemów 30-zadaniowych i o 2,1% mniejsze odchylenie dla problemów 120-zadaniowych). Dodatkowo czas przetwarzania algorytmów stosujących równoległy SGS był średnio 4,2-krotnie dłuższy niż wykorzystujących szeregowy SGS.

W drodze doświadczeń ustalono następujące parametry GA: prawdopodobieństwo krzyżowania 0,6, prawdopodobieństwo mutacji 0,2, liczba osobników elitarnych 2, selekcja metodą rankingową. Wśród metod wyboru nowego pokolenia najlepsze rezultaty osiągano przy zastosowaniu metody rankingowej, ale wpływ wyboru metody selekcji na osiągane wyniki nie był duży.

Zestawienie wyników eksperymentów w zależności od stosowanych operatorów genetycznych dla GA i typów ruchu dla SA umieszczono w tabeli 1. Dla GA przebadano każdą z kombinacji operatora krzyżowania (PPX, 1PX oraz 2PX) i mutacji (Zamień, Hartmann, Wstaw oraz Inwersja). Dla SA przetestowano każdy z typów ruchu (Zamień, Hartmann, Wstaw, Inwersja, Wstaw Wszystkie i Zamień Wszystkie).

Tabela 1

Wyniki eksperymentów obliczeniowych (szeregowy SGS)

Lp.	Parametry algorytmów	30 zadań			120 zadań		
		a	b	c	a	b	C
1	GA, PPX, Zamień	139	170	0,56%	28	98	13,59%
2	GA, PPX, Hartmann	100	131	3,06%	9	39	25,58%
3	GA, PPX, Wstaw	148	170	0,45%	50	156	9,53%
4	GA, PPX, Inwersja	125	154	1,15%	19	70	18,2%

5	GA, 1PX, Zamień	142	163	0,72%	27	83	16,50%
6	GA, 1PX, Hartmann	100	136	2,74%	15	49	22,64%
7	GA, 1PX, Wstaw	153	**173**	**0,23%**	53	154	8,12%
8	GA, 1PX, Inwersja	122	160	1,34%	21	65	17,5%
9	GA, 2PX, Zamień	142	164	1,07%	28	91	14,72%
10	GA, 2PX, Hartmann	109	134	2,72%	13	51	21,67%
11	GA, 2PX, Wstaw	144	168	0,41%	53	152	9,34%
12	GA, 2PX, Inwersja	125	158	1,75%	22	80	18,15%
13	SA, Zamień	140	167	0,98%	83	177	1,97%
14	SA, Hartmann	108	145	2,39%	44	155	7,18%
15	SA, Wstaw	143	171	0,29%	**90**	173	**1,47%**
16	SA, Inwersja	137	162	1,11%	80	175	3,49%
17	SA, Zamień Wszystkie	**157**	**173**	0,48%	79	**176**	3,36%
18	SA, Wstaw Wszystkie	106	145	1,88%	51	162	4,44%
19	SA, losowo typ ruchu	114	154	1,23%	41	148	7,81%
20	*LosoweRCPSP*	102	130	2,52%	19	52	15,34%

a – liczba rozwiązań najlepszych (spośród 180 eksperymentów)
b – liczba rozwiązań lepszych od średniej (spośród 180 eksperymentów)
c – średnie procentowe odchylenie względne od najlepszego uszeregowania

Wnioski z obliczeń:
- lepsza skuteczność algorytmu SA niż GA, zwłaszcza dla problemów większych, złożonych ze 120 zadań;
- znacznie lepsze wyniki osiągane przy użyciu najlepszych algorytmów mataheurystycznych niż przy zastosowaniu procedury generującej losowe rozwiązania *LosoweRCPSP*;
- porównywalne wyniki dla różnych technik wieloargumentowych – operatorów krzyżowania;
- zróżnicowane wyniki dla różnych technik jednoargumentowych – najlepsza mutacja (ruch) typu Wstaw i ruch typu Zamień Wszystkie, najgorsza mutacja (ruch) Hartmanna.

7. Zakończenie

W artykule przedstawiono algorytm genetyczny i algorytm symulowanego wyżarzania dla problemu harmonogramowania projektu z kamieniami milowymi. Wyniki testów obliczeniowych potwierdziły przydatność algorytmów metaheurystycznych dla rozważanego problemu. Najlepsze rezultaty są osiągane przy wykorzystaniu algorytmu symulowanego wyżarzania z zastosowaniem ruchów typu Wstaw i Zamień Wszystko.

Przedmiotem dalszych badań będzie opracowanie nowych, wydajniejszych technik przeszukiwania przestrzeni potencjalnych rozwiązań, dedykowanych dla problemu RCPSP ze zdefiniowanymi terminami realizacji etapów projektu.

BIBLIOGRAFIA

1. Bierwirth C., Mattfeld D.C.: Production Scheduling and Rescheduling with Genetic Algorithms, Evolutionary Computation 7 (1), 1999, s. 1-17.
2. Herroelen W., De Reyck B., Demeulemeester E.: Resource constrained scheduling: a survey of recent developments. Computers and Operations Research 25, 1998, s. 279-302.
3. Kirkpatrick S., Gelatt C.D., Vecchi M.P.: Optimization by simulated annealing. Science 220, 1983, s. 671-680.
4. Klimek M., Łebkowski P.: Miary odporności harmonogramów. Komputerowo Zintegrowane Zarządzanie (red.) R. Knosala, Oficyna Wydawnicza Polskiego Towarzystwa Zarządzania Produkcją, t. I, Opole 2008, s. 569-577.
5. Kolisch R.: Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. European Journal of Operational Research 90, 1996, s. 320-333.
6. Kolisch R., Sprecher A.: PSPLIB – a project scheduling library. European Journal of Operational Research 96, 1997, s. 205–216.
7. Kostrubiec A.: Harmonogramowanie projektów - przegląd modeli. Inżynieria Zarządzania Przedsięwzięciami, Wydawnictwo Politechniki Gdańskiej, Gdańsk 2003, s. 33-52.
8. Kostrubiec A.: Metody generowania sąsiedztwa w metaheurystycznych metodach harmonogramowania projektów. Inżynieria systemów zarządzania. Ilościowe metody wspomagania decyzji w systemach produkcji, Wydawnictwo Politechniki Gdańskiej, Gdańsk 2005, s. 45-57.

Recenzent: Prof. dr hab. inż. Czesław Smutnicki

Abstract

In this paper is described the Resource-Constrained Scheduling Problem (RCPSP) and the assumption of timely execution of the project milestones. It's practical approach, because during the execution of production orders, workers often determine the terms of completion of particular production stages.

For proposed model we have defined the objective function, taking into consideration the protection level of all the project stages and implemented genetic and simulated annealing algorithms. We have tested effectiveness different combinations of parameters, operators (moves) of these metaheuristics. Experiments show that simulated annealing algorithms outperform genetic algorithms. The best results we have achieved using simulated annealing algorithm with moves Insert and Swap All.