ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1998
Seria: AUTOMATYKA z. 125 Nr kol. 1391

Paweł KOMINEK
Politechnika Poznańska

OPERATORY GENETYCZNE + UCZENIE MASZYNOWE = EFEKTYWNE ALGORYTMY EWOLUCYJNE

Streszczenie. W pracy rozważany jest problem szeregowania zadań (RCPS) minimalizujący długość uszeregowania C_{max}. Ze względu na dużą złożoność obliczeniową [2] do jego rozwiązania proponuje się zastosowanie algorytmu genetycznego „wzbogaconego wiedzą" (KAGA). W pracy przedstawione są nowe operatory genetyczne, jak: krzyżowanie i mutacja, uwzględniające sieć ograniczeń kolejnościowych. Ostatni rozdział zawiera porównanie działania GA i KAGA na podstawie eksperymentu obliczeniowego.

GENETIC OPERATORS + MACHINE LEARNING = EFFECTIVE EVOLUTIONARY ALGORITHM

Summary. The paper considers a resource-constraints project-scheduling problem (RCPS) of minimizing the makespan C_{max}. Because of high computational complexity of the problem [2] the use knowledge-augmented of genetic algorithm (KAGA) is proposed. The paper present new genetic operators as: crossover and mutation. The operators which take into account the precedence constraints. The last section includes compare the computationally experimented convergence of the standard genetic algorithm GA with KAGA.

1. Wprowadzenie

Zastosowanie algorytmów metaheurystycznych, tj. symulowanego wyżarzania (SA), przeszukiwania tabu (TS), algorytmu genetycznego (GA) do rozwiązywania problemów NP-trudnych [2] nie jest nowością. W wielu artykułach i rozprawach [3], [8], [9], [10], [11], [12] stwierdzano przydatność tych algorytmów, wykazując ich efektywność i skuteczność. Pierwsze dwa algorytmy SA i TS posiadają mechanizmy pozwalające zawęzić obszar poszukiwań najlepszego rozwiązania. TS - operuje na liście tabu, a SA na parametrze kontrolnym, jakim jest temperatura. Z tego względu SA i TS uchodzą za bardziej skuteczne. GA przeszukuje całą przestrzeń rozwiązań. Interesująca staje się zatem próba wprowadzenia dodatkowych informacji i modyfikacja operatorów genetycznych w celu poprawy efektywności.

Informacja taka pozwoli ukierunkować przeszukiwania oraz przyspieszyć proces poszukiwania najlepszego rozwiązania. Możliwości uwzględnienia wiedzy w sposobie działania operacji genetycznych lub w tworzeniu sąsiedztwa określa się jako „wzbogacanie wiedzą". Do znanych sposobów połączenia wiedzy szczegółowej z mechanizmami algorytmów metaheurystycznych zaliczamy techniki hybrydyzacji i metody aproksymowania funkcji celu [9]. W poprzedniej pracy [8] autor przedstawił definicję sąsiedztwa, która uwzględnia sieć ograniczeń kolejnościowych dla SA i TS. Wyniki, jakie zostały przedstawione, jednoznacznie pozwalają określić wyższość tych algorytmów nad GA. Głównym powodem niskiej efektywności jest fakt, że proste operatory genetyczne (krzyżowanie i mutacja), prowadzące do wymiany zadań pomiędzy chromosomami, nie zapewniają tworzenia nowego rozwiązania [10]. Zachodzi to np. w przypadku wymiany zadań równolegle wykonywanych.

W tej pracy przedstawiony jest nowy sposób przeprowadzenia operacji krzyżowania i mutacji (rozdział 3.1). Polega on na uwzględnieniu sieci ograniczeń kolejnościowych. W następnym rozdziale przedstawiony jest sposób uwzględnienia wiedzy pozyskanej ze zbioru populacji do tworzenia operatorów genetycznych. Wpływ na jakość otrzymanego rezultatu jest treścią rozdziału 4, w którym omówiono eksperyment obliczeniowy, polegający na porównaniu działania standardowego algorytmu genetycznego GA i algorytmu genetycznego wzbogaconego wiedzą KAGA. Podsumowanie pracy zawarte jest w rozdziale ostatnim.

2. Podstawowe pojęcia

2.1. Definicja problemu

Rozpatrywany problem RCPS można sformułować w następujący sposób [14]:

- Projekt zawiera:
  - p typów zasobów odnawialnych należących do zbioru R
  - n zadań ponumerowanych od 1 do n należących do zbioru Z.
- Dla każdego typu zasobu R_s: s = 1,...,p jest określona liczba dostępnych jednostek zasobowych M.
- Dla każdego zadania Z_j: j = 1,...,n jest określony:
  - czas gotowości a_j (Ready Time),
  - czas wykonywania p_j (Proccesing Time),

- oczekiwany czas zakończenia d_j (Due Date),
- wektor żądań zasobowych $r_j = [r_1,...,r_p]$.

- Dla każdego zadania Z_j: $j = 1,...,n$ danych jest $i = 1,...,k$ poprzedników ($k<n$) należących do zbioru P.
- Funkcją kryterialną jest długość uszeregowania C_{max}.

Dla tak sformułowanego problemu rozwiązanie polega na uszeregowaniu wszystkich zadań Z_j: $j = 1,...,n \in Z$ przy zachowaniu:

- ograniczeń kolejnościowych, czyli $\forall Z_j$, $j = 1,...,n$ spełnienie nierówności

$$t_f(P_i) \le t_f(Z_j) - p_j \qquad \forall P_i , \quad i = 1,...,k,$$

- ograniczeń zasobowych, czyli $\forall t$ $t \in <0;C_{max}>$ spełnienie nierówności

$$\sum_{l=1}^{q} r_{j_l} \le M_s \qquad \forall s , \quad s = 1,...,p$$

w celu osiągnięcia minimalnej wartości C_{max},

gdzie: $t_f(P_i)$ - moment zakończenia zadania P_i,

$t_f(Z_j)$ - moment zakończenia zadania Z_j,

q - liczba zadań równolegle wykonywanych.

2.2. Elementarny AG

Procedura algorytmu genetycznego, której schemat przedstawiono na rys.1, prezentowana jest w wielu pracach [3], [9], [10]. Dla problemu zdefiniowanego w pkt. 2.1 możemy stwierdzić, że:

- rozwiązaniem jest ciąg zwany chromosomem,
- w chromosomie znajdują się zadania (geny) uporządkowane wg priorytetu,
- numer zadania to "allele", a pozycja zadania w chromosomie to „locus"[9],
- zastosowana operacja krzyżowania typu LOX (local ordering crossover) polega na wymianie losowo wybranych genów pomiędzy dwoma chromosomami,
- mutacja MUT polega na zamianie miejscami dwóch losowo wybranych genów w chromosomie.

Rys.1.Schemat standardowego algorytmu genetycznego
Fig.1. A Schema of standard genetic algorithm

3. Przykład RCPS

Rozpatrzmy następujący przykład:

Niech dane będą zbiory:

$$Z = [1,2,3,4,5,6],$$

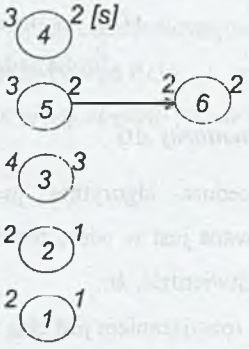
$$R = [5]$$

Sieć ograniczeń kolejnościowych, w której wierzchołki reprezentują zadania, a łuki zależności czasowe przedstawiono na rys.2.

Żądania zasobowe i czasy wykonywania poszczególnych zadań wynoszą odpowiednio:

$r_{js} = [2,2,4,3,3,2]$, dla j=1,...,6 , s=1,

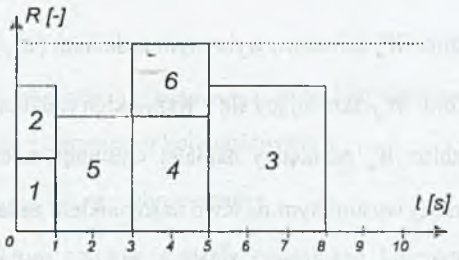
$p_j = [1,1,3,2,2,2]$, dla j=1,...,6.



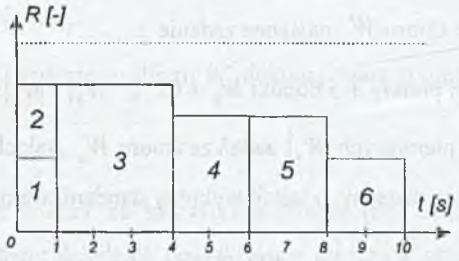
Rys.2. Sieć ograniczeń kolejnościowych
Fig.2. Network of precedence constraints

Przeprowadźmy operację krzyżowania LOX na dwóch chromosomach A, B, którym odpowiada wykres Gantta.

A = 1 2 5 4 6 3



B = 1 2 3 4 5 6



W rezultacie otrzymamy chromosomy A', B':

A= 1 2 5 [4 6] 3
B= 1 2 3 [4 5] 6

po uporządkowaniu

A'= 1 2 6 4 5 3 ⇒ A'= 1 2 4 5 6 3
B'= 1 2 3 4 6 5 ⇒ B'= 1 2 3 4 5 6

Można zauważyć, że taka operacja nie uwzględnia sieci ograniczeń kolejnościowych. Powstają dzieci, które są identyczne z rodzicami. Prowadzi to do niepotrzebnych strat czasu na uporządkowywanie i szeregowanie, w rezultacie których cały algorytm może być mało efektywny.

3.1. Definicja operatora krzyżowania KALOX (wzbogaconego wiedzą)

Przyjęto następujące oznaczenia:

W_A - zbiór genów do wymiany z chromosomu A,

W_B - zbiór genów do wymiany z chromosomu B.

W_B' - podzbiór zbioru W_B,

$|W_A|$ - liczność zbioru W_A,

$|W_B|$ - liczność zbioru W_B,

z_{jA} - zadanie z chromosomu A znajdujące się na pozycji j, $j \in [1,n]$,

1. Utwórz zbiór W_A z losowo wybranym zadaniem (z_{jA}) z chromosomu A.
2. Utwórz zbiór W_B składający się z wszystkich zadań z chromosomu B.
3. Utwórz zbiór W_B'pomiędzy najdalej wysuniętym na prawo poprzednikiem zadania z_{jA} i najdalej wysuniętym na lewo następnikiem zadania z_{jA}.
4. Zmodyfikuj zbiór $W_B := W_B \cap W_B'$.
5. Dodaj do zbioru W_A następne zadanie $z_{(j+1)A}$.
6. Powtarzaj punkty 3-5 dopóki $W_B \neq \emptyset$ i $|W_A| < |W_B|$.
7. Wybierz pierwszych $|W_A|$ zadań ze zbioru W_B, takich że $W_A \neq W_B$.
8. Dla tak wyznaczonych sekcji wykonaj standardową operację krzyżowania LOX.

Rozważmy ponownie przykład 1.

iteracja 1

$z_{jA} = 4$,

A= 1 2 5 [4] 6 3 $W_A = \{4\}$

B= [1 2 3 4 5 6] $W_B^{(1)} = \{1,2,3,4,5,6\}$

iteracja 2

A= 1 2 5 [4 6] 3 $W_A = \{4,6\}$

B= 1 2 3 4 5 [6] $W_B^{(2)} = \{6\}$

Podczas drugiej iteracji wystąpił warunek stopu $|W_A| < |W_B|$. Ostatecznie możemy wymienić zadanie 4 z dowolnym zadaniem znajdującym się w zbiorze W_B pod warunkiem $W_A \neq W_B$. Np.:

W_A = { 4 } W_B = { 1 }

Po przeprowadzeniu operacji krzyżowania LOX mamy: A'= 2 5 4 1 6 3

B'= 4 1 2 3 5 6

W wyniku przeprowadzonej operacji krzyżowania KALOX powstałe potomstwo zawsze będzie różnić się od rodziców.

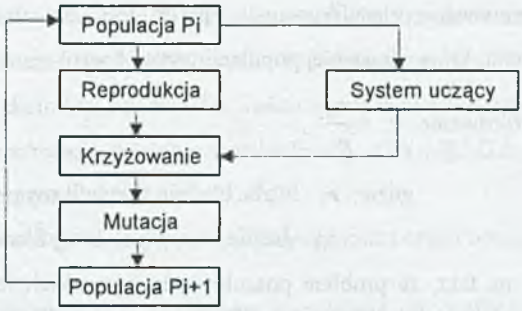
3.2. Definicja mutacji KAMUT

Opierając się na definicji operatora krzyżowania KALOX, możemy zdefiniować operację mutacji uwzględniającą sieć ograniczeń kolejnościowych:

1. Wybierz losowo zadanie z_j, $j \in [1,n]$ z chromosomu.
2. Utwórz zbiór W zadań, które zgodnie z siecią ograniczeń kolejnościowych mogą zostać wymienione z zadaniem z_j.
3. Wykorzystując zadanie z_j i zadanie ze zbioru W dokonaj operacji mutacji.

3.3. Ekstrakcja wiedzy - KAGA

Oprócz technik pozyskiwania wiedzy ze specyfiki problemu (np. uwzględnianie sieci ograniczeń kolejnościowych) istnieje dziedzina pozyskiwania wiedzy z przykładów, zwana uczeniem maszynowym. Interesujące wydaje się zastosowanie tej dziedziny w celu lepszego poznania analizowanego przykładu a tym samym skonstruowania efektywniejszych operatorów genetycznych. Poniżej zaprezentowany jest schemat blokowy KAGA, czyli GA, zawierający elementy uczenia maszynowego.

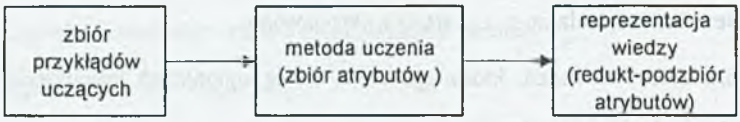


Rys.3. Schemat algorytmu genetycznego wzbogaconego wiedzą
Fig.3. A Schema of knowledge-augmented of genetic algorithm

Danymi wejściowymi do systemu uczącego jest populacja N chromosomów. W zależności od wartości funkcji dopasowania każdy chromosom zostaje przydzielony do jednej z dwóch klas D, Z. Klasę D o liczności $10\%N$ stanowią najlepsze chromosomy z populacji. Reszta należy do klasy Z.

Pozycje genów w chromosomie stanowią atrybuty. Zadaniem systemu uczącego jest znalezienie takich atrybutów, które przyczyniają się do jednoznacznego zaklasyfikowania

każdego chromosomu do jednej z wymienionych klas, a następnie do zredukowania liczby atrybutów do minimalnego podzbioru. Schemat blokowy takiego systemu uczącego przedstawia rys. 4.



Rys.4. Schemat sytemu uczącego
Fig.4. A schema of knowledge system

Otrzymany zbiór atrybutów można wykorzystać w pkt. 1. definicji operatora krzyżowania KAGA lub operatora mutacji KAM.

Dla poprawnego i efektywnego działania systemu uczącego przyjęto następujące założenia:

- redukcja atrybutów jest poszukiwaniem zredukowanego minimalnego podzbioru atrybutów, zapewniającego tę samą jakość klasyfikacji jak oryginalny zbiór atrybutów,
- ocena poprawności klasyfikowania oparta jest na oszacowaniu tzw. błędu klasyfikowania, który dla każdej populacji wynosił zero;

  - błąd klasyfikowania $e_r = \frac{n_e}{n_N}$,

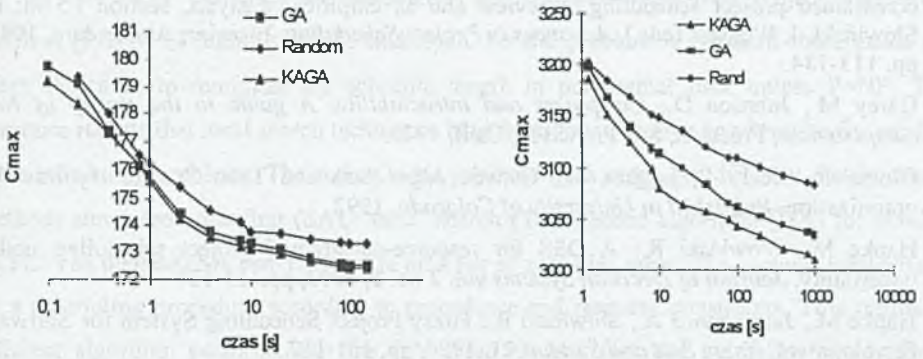
  gdzie: n_e - liczba błędnie sklasyfikowanych obiektów,

  n_N - liczba wszystkich klasyfikowanych obiektów.
- ze względu na fakt, że problem poszukiwania wszystkich reduktów jest NP-trudny, w pracy ograniczono się do wygenerowania dla każdej populacji jednego reduktu.

4. Eksperyment obliczeniowy; porównanie GA i KAGA

Eksperyment obliczeniowy został przeprowadzony w Poznańskim Centrum Superkomputerowo-Sieciowym na superkomputerze SGI Power (8 procesorów R8000 75 MHz, 300 Mflops, 512 MB RAM, 16GB pamięci dyskowej, systemem operacyjnym UNIX w wersji IRIX 6.0).

Każdy punkt na wykresie jest średnią wartością 50 niezależnych testów poszczególnych algorytmów.



Rys. 5. Wykres średniej wartości Cmax w funkcji czasu obliczeniowego dla problemu „40 i 80 zadań"
Fig. 5.Chart of mean value of Cmax as a function of computation time for the problem of „40 and 80 activities"

5. Podsumowanie

Przedstawiona praca zawiera zdefiniowaną operację krzyżowania i mutacji, które uwzględniają sieć ograniczeń kolejnościowych. Przedstawiony został również algorytm genetyczny z modułem ekstrakcji wiedzy. Na końcu zamieszczone są wyniki eksperymentu obliczeniowego, które porównują działanie standardowego GA i KAGA. Z rezultatów tych można wyciągnąć następujące wnioski:

- koszt czasowy ekstrakcji wiedzy jest mniejszy niż koszt czasu tracony na powtarzające się uszeregowania,
- w każdym z rozpatrywanych problemów niezależnie od wielkości instancji KAGA wykazuje wyższość nad GA,
- dla problemów o dużej instancji różnice te są jeszcze wyższe.

Podziękowania

Praca ta została wykonana w ramach Działalności Statutowej oraz projektu badawczego KBN Nr 8 T11C 013 13 i grantu CRIT II. Obliczenia wykonano na zasobach Poznańskiego Centrum Superkomputerowo-Sieciowego.

Autor pragnie również podziękować dr. inż. A. Jaszkiewiczowi za cenne uwagi metodologiczne.

LITERATURA

1. Alvares-Valdés Olaguíbel R., Tamarit Goerlich J.M.: Heuristic algorithms for resource-constrained project scheduling: a review and an empirical analysis, section I.5 in: R. Słowiński, J. Węglarz (eds.), *Advances in Project Scheduling*, Elsevier, Amsterdam, 1989, pp. 113-134.
2. Garey M., Johnson D.: *Computers and intractability: A guide to the theory of NP-completeness*, Freeman, San Francisco, Calif, 1979.
3. Glover F., Keely J., Laguna M.: Genetic Algorithms and Tabu Search: Hybrids for optimization. *Published in University of Colorado*, 1992.
4. Hapke M., Słowiński R.: A DSS for resource-constrained project scheduling under uncertainty, *Journal of Decision Systems* vol. 2 no. 2, 1993, pp.111-128.
5. Hapke M., Jaszkiewicz A., Słowiński R.: Fuzzy Project Scheduling System for Software Development, *Fuzzy Sets and Systems* 21, 1994, pp. 101-117.
6. Hapke M.: Two-stage fuzzy optimization for project scheduling, *Proceedings of the Operational Research of Italy Annual Conference, AIRO'95*, Ancona, 20-22 September 1995, pp. 295-298.
7. Hapke M.: Programowanie sieciowe w warunkach niepewności, *Praca przedłożona do publikacji w Zeszytach Naukowych Politechniki Śląskiej, prezentowana na X KKADPP, Kozubnik, 18-21 września 1996.*
8. Hapke M., Kominek P., Słowiński R., *Metaheurystyczne procedury rozwiązywania problemu programowania sieciowego w warunkach niepewności*, Zeszyty Naukowe Politechniki Śląskiej, Automatyka z. 117, Gliwice, s. 57-68, 1996.
9. Goldberg D.: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.
10. Kominek P.: *Efektywność algorytmu genetycznego w zastosowaniu do problemów rozmytego programowania sieciowego*, Praca publikowana w materiałach I Krajowej Konferencji Algorytmy Ewolucyjne, Murzasichle 1996.
11. Laarhoven P.J.M., Aarts E.H.L., *Simulated Annealing: Theory and Practice*, Kluwer Academic Publishers, Dordreht, The Netherlands, 1987.
12. Laguna M., Barnes J.W., Glover F.W.: *Tabu search methods for a single machine scheduling problem*, Journal of Intelligent Manufacturing, 2, 63-74, 1991.
13. Lawrence S.R.: *An experimental investigation of heuristic scheduling techniques, GSIA*, Carnegie-Mellon University, Pittsburgh, 1984.
14. Patterson J.H., Słowiński R., Talbot F.B., Węglarz J.: *An algorithm for a general class of precedence and resource constrained scheduling problems.* section I.1 in: Słowiński R., Węglarz J. (eds.), *Advances in Project Scheduling*, Elsevier, Amsterdam, pp. 3-28, 1989.
15. Stefanowski J.: *Dialogowe wspomaganie decyzji na podstawie wiedzy pozyskanej metodą zbiorów przybliżonych*. Rozprawa doktorska wykonana pod kierunkiem prof. dr hab. inż. Romana Słowińskiego, Instytut Informatyki Politechniki Poznańskiej, Poznań 1994.

Recenzent: Prof.dr hab.inż. Jerzy Klamka

Abstract

In this paper we study the precedence and resource-constrained project scheduling problem (RCPS) of minimizing the makespan. As the problem is NP-hard there exists no exact algorithm to minimize the schedule length in polynomial time unless P=NP. The literature reports that local search techniques have been found to give good results for solving these scheduling problems. The author in his previous work compared different metaheuristic methods simulated annealing (SA), "tabu" search (TS), genetic algorithm (GA) for solving RCPS. The methods SA and TS operate on a list of activities taken into account sequentially by a scheduling procedure according to precedence and resource constraints. This results in efficient algorithm performance. The GA based method that had not taken into account constraints did not give as good results as SA and TS. It performed „better" only for problems of huge instances. In this paper we are motivated to present new genetic operators for algorithm solving RCPS problems. The operators produce children that are different from their parents and find better solutions faster. New schedules are constructed by changing the order of the list. A simple exchange of activities positions may not lead to new schedules. Such situation would occur if positions of two activities, performed in the previous schedule in parallel, had been exchanged.

The paper is show haw good definition of genetic operators (crossover, mutation) influences the calculation. In the last section author compares the computationally experimented convergence of the GA with effective genetic operators.