I Krajowa Konferencja
Algorytmy Ewolucyjne
Murzasichle, 12-15.06.1996

Paweł KOMINEK
Instytut Informatyki Politechniki Poznańskiej

EFEKTYWNOŚĆ ALGORYTMU GENETYCZNEGO W ZASTOSOWANIU DO PROBLEMÓW ROZMYTEGO PROGRAMOWANIA SIECIOWEGO

1. Wprowadzenie

W pracy rozpatrywany jest problem programowania sieciowego w warunkach niepewności. Niepewność ta związana jest z niemożliwością precyzyjnego określenia parametrów czasowych poszczególnych operacji i wyrażana jest za pomocą liczb rozmytych. W ogólności problem rozmytego programowania sieciowego polega na takim rozdziale zasobów do zadań, aby zachowując wszystkie ograniczenia (zasobowe i kolejnościowe) osiągnąć jak najlepszą wartość kryterium optymalizacji. Należy tu podkreślić, że wszystkie parametry typu czasowego poszczególnych zadań oraz kryterium maksymalnej długości uszeregowania są niepewne i będą w tej pracy wyrażane w kategoriach liczb rozmytych. Szczegółowe sformułowanie problemu zawarte zostało w pracy [7].

Ze względu na dużą złożoność obliczeniową problemu - problem NP-trudny [2], do jego rozwiązywania można zastosować metody dokładne, specjalizowane heurystyki lub metaheurystyki. Metody dokładne (np. metoda podziału i ograniczeń [12]) dają oczywiście rozwiązania optymalne, ale nie mogą być stosowane do rozwiązywania problemów o realnych rozmiarach ze względu na długi czas obliczeń. Specjalizowane heurystyki (np. heurystyki priorytetowe [1,11]) są zwykle bardzo efektywne i dają niezłe rozwiązania. Dużo lepsze rozwiązania można jednak uzyskać stosując procedury metaheurystyczne (symulowana relaksacja (SA), przeszukiwanie tabu (TS) lub algorytm genetyczny (GA)). Choć obliczenia metaheurystyczne trwają zwykle dłużej od tych, które wykorzystują heurystyki specjalizowane, charakteryzują się one dobrą zbieżnością do rozwiązań bliskich rozwiązaniu optymalnemu.

Tematem tej pracy jest omówienie w zarysie procedury algorytmu genetycznego oraz jego adaptację w połączeniu z rozmytą procedurą szeregującą [7] do rozwiązywania rozpatrywanego problemu. Następnie omówienie eksperymentu obliczeniowego porównującego działanie procedur metaheurystycznych: symulowanej relaksacji,

przeszukiwania tabu i algorytmu genetycznego. Podsumowanie pracy zawarte jest w rozdziale czwartym.

2. Algorytm genetyczny - zarys metod

Działanie tego algorytmu nawiązuje do mechanizmu występującego w przyrodzie jakim jest naturalna selekcja genetyczna i oparte jest na populacji rozwiązań. Rozwiązania te współdziałają ze sobą, mieszają się tworząc nową generację – „dzieci", które dziedziczą dobre cechy po „rodzicach". Algorytm genetyczny możne więc być opisany jako mechanizm naśladujący ewolucję genetyczną gatunku w kierunku uzyskania wybranych cech. Głównymi operatorami odpowiedzialnymi za utworzenie nowej generacji są operatory reprodukcji, krzyżowania oraz mutacji. Literatura dotycząca algorytmu genetycznego bogata jest w terminologię zaczerpniętą z genetyki [9]. Gdy mówimy „chromosom" mamy na myśli zakodowane rozwiązanie. Elementami chromosomu są „geny". Wartość „genu" to „allele", a pozycja w rozwiązaniu to inaczej „locus". Ponadto z każdym chromosomem związana jest pewna wartość zwana dopasowaniem, której odpowiada wartość funkcji kryterialnej.

Algorytm genetyczny rozpoczyna działanie od utworzenia populacji N rozwiązań. Rozmiar populacji pozostaje stały przez cały czas działania algorytmu. Generacja $n+1$ tworzona jest z n-tej generacji $X(n)$ w następujący sposób. Z generacji $X(n)$ na podstawie dopasowania wybierane są najlepsze osobniki. Następnie dokonywane są na nich operacje krzyżowania czego rezultatem jest powstanie potomstwa, które zastąpi złych osobników w bieżącej populacji. Dalej na małej grupce „dzieci" dokonywana jest operacja mutacji. Algorytm genetyczny może być zapisany w następujący sposób:

```
procedure ALGORYTM_GENETYCZNY;
begin
  INICJALIZUJ(X(1));
  for i:=2 to K do
  begin
    WYBIERZ_DOBRYCH_OSOBNIKÓW; {z X(i)}
    KRZYŻOWANIE;
    MUTACJA;
    ZASTĄP_ZŁYCH_OSOBNIKÓW;
  end;
end;
```

gdzie K jest ustaloną liczbą iteracji.

W rozpatrywanym problemie chromosomy będą odpowiadać listom priorytetowym, a geny zadaniom. Każda lista priorytetowa składa się zatem z zadań (uporządkowanych w kolejności malejących priorytetów) a jej rozmiar określa ilość zadań do wykonania. Główne operacje odpowiedzialne za utworzenie nowej generacji: reprodukcja, krzyżowanie oraz mutacja, zmieniają kod genetyczny chromosomu tak bardzo, że istnieje tylko bardzo małe

prawdopodobieństwo, że dla takiego sposobu kodowania rozwiązania, dzieci będą identyczne z rodzicami. Poniżej opisana jest adaptację tych trzech kluczowych operacji do rozważanego tu problemu programowania sieciowego.

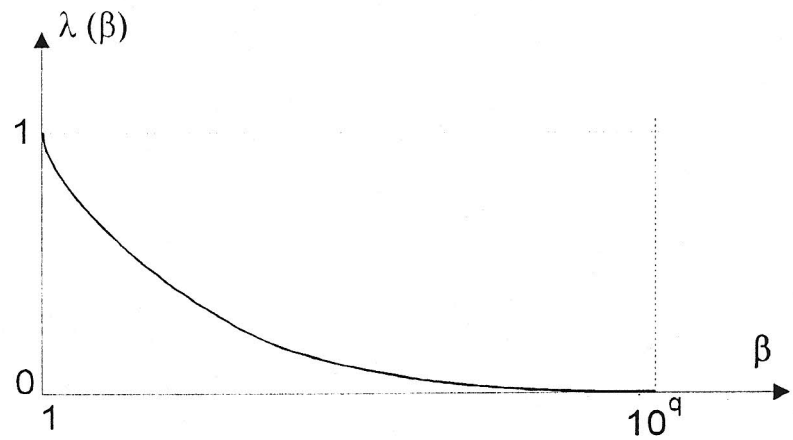
Reprodukcja polega na losowym wyborze chromosomów do stworzenia nowej generacji. Im lepsze dopasowanie tym większe prawdopodobieństwo wyboru chromosomu. Proponuje się, by chromosom *n1* został wybrany, gdy $\lambda_{n1}>\lambda_{n2}$:

gdzie: $\lambda_{n1} = 1 - \frac{\sqrt[q]{\beta}}{10}$ (rys. 1), $\lambda_{n2} = \frac{F(x_i)-F_{min}}{F_{max}-F_{min}}$,

β – wartość losowa z przedziału [1 ; 10^q],

q – stopień preferencji dobierany doświadczalnie.

F_{min}, F_{max} – najlepsze i najgorsze dopasowania chromosomu,



Rys. 1. Wykres zmian współczynnika λ_{n1} w zależności od parametru β.

Z pośród wielu operacji **krzyżowania** zastosowana tutaj operacja LOX (krzyżowanie lokalno porządkowe [9]) polega na wyszukaniu genów występujących w granicach krzyżowania jednego chromosomu w drugim chromosomie i odwrotnie. Następnie wyszukane geny usuwane są z chromosomów, a puste miejsca przesuwa się tak, by znalazły się w granicach krzyżowania. W następnym kroku następuje wymiana zaznaczonych fragmentów kodu. Dla większej przejrzystości działanie tej operacji zaprezentujemy na przykładzie chromosomu zawierającego osiem genów (lista priorytetowa ośmiu zadań).

Niech dane będą dwa chromosomy :

A = 4 1 6 8 2 7 3 5 B = 3 7 2 4 5 1 6 8

Granice krzyżowania wylosowano dla pozycji 3 i 5. W chromosomach A^1 i B^1 wybrano, a w A^2 i B^2 usunięto te geny, które znalazły się w granicach krzyżowania chromosomów B i A odpowiednio.

$A = 4\ 1\ [6\ 8\ 2]\ 7\ 3\ 5$ $\quad A^1 = \underline{4}\ 1\ [6\ 8\ \underline{2}]\ 7\ 3\ \underline{5}$

$B = 3\ 7\ [2\ 4\ 5]\ 1\ 6\ 8$ $\quad B^1 = 3\ 7\ [\underline{2}\ 4\ 5]\ 1\ \underline{6}\ \underline{8}$

$A^2 = H\ 1\ [6\ 8\ H]\ 7\ 3\ H$ $\quad A^3 = 1\ 6\ [H\ H\ H]\ 8\ 7\ 3$

$B^2 = 3\ 7\ [H\ 4\ 5]\ 1\ H\ H$ $\quad B^3 = 3\ 7\ [H\ H\ H]\ 4\ 5\ 1$

Po przesunięciu pustych miejsc (A^3 i B^3) dokonano wymiany fragmentów kodu i otrzymano:

$A' = 1\ 6\ 2\ 4\ 5\ 8\ 7\ 3$

$B' = 3\ 7\ 6\ 8\ 2\ 4\ 5\ 1$

Operacja **mutacji** jest wykonywana na pojedynczym chromosomie i polega na zamianie miejscami dwóch losowo wybranych genów.

3. Eksperyment obliczeniowy: porównanie algorytmu genetycznego, symulowanej relaksacji i przeszukiwania tabu

Eksperyment obliczeniowy został przeprowadzony w Poznańskim Centrum Superkomputerowo-Sieciowym na superkomputerze SGI Power Challenge (8 procesorów R8000 75 MHz o mocy obliczeniowej 300 MFlops każdy, 512 MB pamięci operacyjnej, 16 GB pamięci dyskowej, pod systemem operacyjnym UNIX w wersji IRIX 6.0).

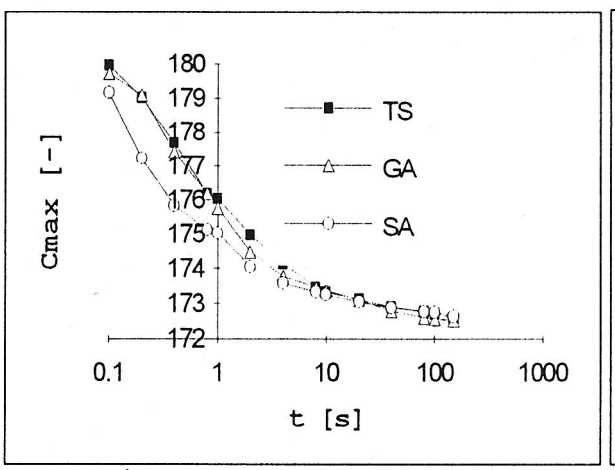
Doświadczenia przeprowadzono przy zastosowaniu rozmytej procedury szeregującej [7] na problemach składających się z 40, 60, 80, i 100 zadań. Problemy te zostały wygenerowane losowo i nieznane są dla nich optymalne wartości maksymalnej długości uszeregowania. Dlatego wyniki przedstawione są w odniesieniu do najlepszych jak dotąd uzyskanych rezultatów optymalizacji. Wynikiem optymalizacji rozmytej jest zawsze rezultat rozmyty. Porównywanie poszczególnych rezultatów rozmytych sprowadza się, jednak, w naszym przypadku do porównania liczb rzeczywistych odpowiadających środkom ciężkości liczb rozmytych. Dla większej przejrzystości wszystkie rezultaty optymalizacji będą tu zatem wyrażane w takiej właśnie postaci.

Eksperyment przeprowadzono w dwóch etapach. Pierwszy polegał na wyznaczeniu jak najlepszych parametrów dla każdej z badanych metaheurystyk [8]. Dla algorytmu genetycznego dobrano następujące parametry:

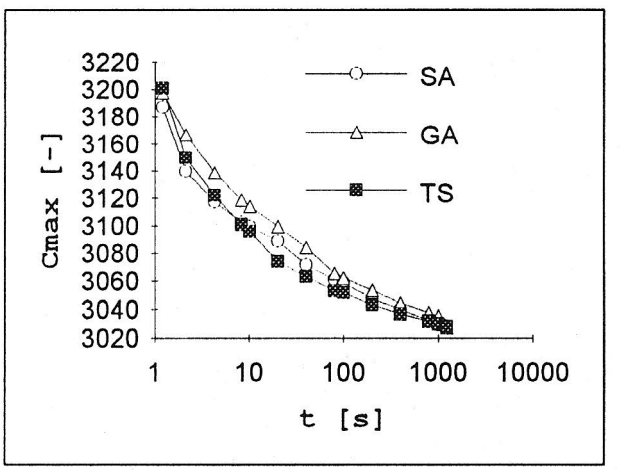
rozmiar pokolenia	$H=0.5*N$
liczba pokoleń	$K=140*N$
prawdopodob. krzyżowania	$P_k=0.6$
prawdopodob. mutacji	$P_m=0.08$
stopień preferencji chromosomu przy reprodukcji	$q=4$

gdzie N jest liczbą zadań w problemie.

Drugi etap polegał na porównaniu działania procedur metaheurystycznych. Przeprowadzono 50 niezależnych prób rozwiązania tego samego problemu. W każdej próbie dyskretyzowano czas obliczeń zapamiętując dwie wielkości: wartość funkcji kryterialnej oraz czas, który upłynął od początku obliczeń. Po wykonaniu wszystkich prób dla każdej chwili czasowej obliczono średnią wartość funkcji kryterialnej. Rysunki 3 i 4 przedstawiają średnią wartości maksymalnej długości uszeregowania C_{max} w funkcji czasu obliczeń dla trzech omawianych procedur metaheurystycznych rozwiązujących problemy składające się z 40 i 80 zadań.



Rys. 2. *Średnia wartość C_{max} w funkcji czasu obliczeń dla problemu "40 zadań".*



Rys. 3. *Średnia wartość C_{max} w funkcji czasu obliczeń dla problemu "80 zadań"*

Dla małych problemów, o wielkości ok. 40 zadań (rys. 2), można zauważyć, że wszystkie rozważane procedury metaheurystyczne osiągają zbliżone rezultaty już po 10 s. Są one oddalone o 1% od wartości Cmax = 172.4, gdzie Cmax = 172.4 jest najlepszym uzyskanym do tej pory rezultatem kryterium długości uszeregowania.

W przypadku problemów o większych rozmiarach (rys. 3), pierwsza faza obliczeń ukazuje przewagę algorytmów symulowanej relaksacji i przeszukiwania tabu. Algorytm genetyczny osiąga dobre rezultaty zbliżone do SA i TS, ale po dłuższym czasie obliczeń. Powodem tego jest zdolność algorytmu genetycznego do dokładnego przeszukiwania przestrzeni rozwiązań dopuszczalnych.

4. Podsumowanie

W pracy zaproponowano rozwiązanie problemu rozmytego programowania sieciowego za pomocą procedur metaheurystycznych: symulowanej relaksacji, przeszukiwania tabu oraz algorytmu genetycznego. Przedstawiono zarys procedury algorytmu genetycznego oraz jego adaptację do rozwiązywania rozpatrywanego problemu. W pracy zaprezentowano sposób

efektywnego konstruowania operatorów: reprodukcji, krzyżowania i mutacji charakterystycznych dla algorytmu genetycznego.

Omówiono eksperyment obliczeniowy, który polegał na wyznaczeniu najlepszych parametrów procedur metaheurystycznych oraz na porównaniu wyników działania tych procedur na wybranych problemach. Wyniki eksperymentu pokazują, że najlepsze wyniki obliczeń dla mniejszych problemów daje procedura symulowanej relaksacji. Lepsze rezultaty dla problemów większych były osiągane przy użyciu procedury przeszukiwania tabu. Algorytm genetyczny osiąga rezultaty równie dobre, jednak po dłuższym czasie obliczeń.

LITERATURA:

[1] R. Alvares-Valdés Olaguíbel, J.M. Tamarit Goerlich: 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] M. Garey, D. Johnson: *Computers and intractability: A guide to the theory of NP-completeness*, Freeman, San Francisco, Calif, 1979.

[3] F. Glover, J. Keely, M. Laguna: Genetic Algorithms and Tabu Search: Hybrids for optimization. *Published in University of Colorado,* 1992.

[4] M. Hapke, R. Słowiński: A DSS for resource-constrained project scheduling under uncertainty, *Journal of Decision Systems* vol. 2 no. **2**, 1993, pp.111-128.

[5] M. Hapke, A. Jaszkiewicz, R. Słowiński: Fuzzy Project Scheduling System for Software Development, *Fuzzy Sets and Systems* **21**, 1994, pp. 101-117.

[6] M. Hapke: 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] M. Hapke: Programowanie sieciowe w warunkach niepewności, *Praca przedłożona do publikacji w Zeszytach Naukowych Politechniki Śląskiej, prezentowana na X KKADPP, Kozubnik, 18-21 września 1996.*

[8] M. Hapke, P. Kominek, R. Słowiński: Metaheurystyczne procedury rozwiązywania problemu programowania sieciowego w warunkach niepewności, *Praca przedłożona do publikacji w Zeszytach Naukowych Politechniki Śląskiej, prezentowana na X KKADPP, Kozubnik, 18-21 września 1996.*

[9] D. Goldberg: *Genetic Algorithms in Search, Optimization and Machine Learning,* Addison-Wesley, 1989.

[10] P.J.M. van Laarhoven, E.H.L. Aarts, *Simulated Annealing: Theory and Practice*, Kluwer Academic Publishers, Dordreht, The Netherlands, 1987.

[11] S.R. Lawrence: An experimental investigation of heuristic scheduling techniques, *GSIA*, Carnegie-Mellon University, Pittsburgh, 1984.

[12] J.H. Patterson, R. Słowiński, F.B. Talbot., J. Węglarz: An algorithm for a general class of precedence and resource constrained scheduling problems. section I.1 in: R. Słowiński, J. Węglarz (eds.), *Advances in Project Scheduling*, Elsevier, Amsterdam, pp. 3-28, 1989.