**Projektowanie Efektywnych Algorytmów**

Jednoprocesorowy problem szeregowania zadań

Algorytm Genetyczny

Autor: Imie Nazwisko

Nr indeksu:

Termin oddania:

Prowadzący:

Termin zajęć:

Kierunek: Informatyka

1. **Wstęp teoretyczny**

Pełna nazwa problemu to „Jednoprocesorowy problem szeregowania zadań przy kryterium minimalizacji ważonej sumy opóźnień zadań”.

Danych jest *n* zadań, które mają być wykonane bez przerwań przez pojedynczy procesor, mogący wykonywać co najwyżej jedno zadanie jednocześnie. Każde zadanie jest dostępne do wykonania w chwili zero, do wykonania wymaga pi > 0 jednostek czasu, ma określoną wagę wi > 0 i oczekiwany termin zakończenia wykonywania di > 0. Zadanie *i* jest spóźnione, jeżeli zakończy się wykonywać po swoim terminie d*i*, miarą tego opóźnienia jest Ti = max(0, Ci – di), gdzie Ci to termin zakończenia zadania *i*. Problem polega na znalezieniu takiej kolejności zadań aby zminimalizować kryterium .

1. **Sposób przeprowadzenia eksperymentu**
   1. **Metoda**

Algorytm Genetyczny (GA) wykorzystywany jest do otrzymywania wyników optymalnych, lub niewiele różniących się od optymalnych dla problemów optymalizacyjnych. Ideą algorytmu jest tworzenie populacji rozwiązań i jej modyfikacji dzięki zastosowaniu operatorów genetycznych np. krzyżowanie, mutacja. Po tym dokonywana jest selekcja, aby wyróżnić najlepsze osobniki które nadają się później do reprodukcji. Każda generacja jest sortowana według wyniku, pierwszy osobnik więc jest najlepszym rozwiązaniem w skali globalnej.

Krzyżowanie polega na łączeniu cech pochodzących z różnych osobników populacji w celu polepszenia populacji. Może się zdarzyć, że odpowiednia kombinacja cech rodzica okaże się lepsza niż u któregoś z rodziców i przyczynia się to do tworzenia następnych generacji z lepszymi cechami. Dla użytego tutaj algorytmu krzyżowanie polega na wylosowanie sekcji dopasowania i przeniesienia kawałków chromosomów w odpowiednie miejsca u dzieci i wypełnienie pozostałej części cech dzieci cechami rodziców.

Ze względu na specyfikę problemu należy zwrócić uwagę, że kopiowanie bezpośrednio może spowodować duplikaty cech. Unika się tego poprzez sprawdzenie czy w pozostałej części chromosomu nie znajduje się już cecha którą zamierza dziecko odziedziczyć po rodzicach.

Algorytm genetyczny może często generować osobniki które powodują, że algorytm staje się zbieżny, czyli wiele osobników ma ten sam zestaw cech, który jest coraz bardziej powielany. Nie jest to pożądane, ponieważ operowanie na tych samych zestawach nie pozwala na znalezienie nowych rozwiązań. Mutacja pozwala temu zapobiec (przynajmniej częściowo).

Mutacja polega na wprowadzeniu do genotypu losowych zmian. Dzięki niej można wprowadzić większą różnorodność w populacji. Zachodzi ona z pewnym założonym prawdopodobieństwem. Im wyższe prawdopodobieństwo, tym częstsze mutacje. Jeśli będą występować zbyt często, możliwe, że ucierpią też widocznie na tym dobre rozwiązania, ponieważ będą one w ten sposób niszczone.

Nowa generacja jest wybierana na podstawie wyniku sortowania sumy grup osobników rodzicielskich i dzieci. Najlepsza połowa dla której wynik funkcji celu był wyższy niż dla pozostałej części staje się nową populacją.

Algorytm kończy się po wygenerowaniu ustalonej liczby generacji. Pierwszy osobnik z ostatniej generacji jest najlepszym ze wszystkich znalezionych osobników na przestrzeni wszystkich generacji.

* 1. **Założenia**
     1. Program testowy jest wykonany jako aplikacja konsolowa
     2. Aby móc porównać z wynikami uzyskanymi za pomocą algorytmu programowania dynamicznego (DPA) i metodą Tabu Seach (TSM), wykonano badanie dla takich samych wielkości problemu (badanie 1)
        + Wielkość populacji to 4 \* ilośćZadań (populacja powinna być parzysta)
        + Ilość iteracji to 2 \* ilośćZadań
        + Szansa mutacji to stałe 5%
        + Badania zostały wykonane dla 7 wartości N zadań odpowiadającym tym wykorzystanym w przypadku DPA
        + Dla każdej wartości N wygenerowano 50 losowych instancji problemu (w sprawozdaniu podano tylko wyniki uśrednione)
     3. Implementacja została wykonano obiektowo
     4. Funkcja celu w tym algorytmie to wyliczenie kosztu sumy opóźnień ważonej zadań. Im mniejszy koszt, tym wyższa wartość funkcji.
     5. Sprawdzono też jak bardzo odbiegał wynik GA od wyniku właściwego dla dużych instancji (badanie 2), porównano to też z TSM z projektu 2.
        + Instancje mają wielkości 40, 50 i 100
        + Dla każdej wielkości instancji wykonano po 10 testów dla 3 różnych zestawów
        + Wynik został uśredniony
        + Parametry zostały dopasowane jak w badaniu 1

1. **Wyniki**
2. Porównanie TSM, DPA i GA

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ilość zadań [n] | 6 | 9 | 12 | 15 | 18 | 21 | 24 | Algorytm |
| Średni czas wykonania [ms] | 0,08096 | 0,88695 | 8,94367 | 91,4503 | 920,358 | 9260,12 | 89113,9 | DPA |
| 0,05071 | 0,14537 | 0,34673 | 0,69347 | 1,04194 | 1,55352 | 2,35199 | TSM |
| 3,09034 | 10,1449 | 25,1073 | 51,5945 | 100,66 | 151,434 | 231,021 | GA |

Tabela 1. Porównanie średniego czasu wykonania dla ilości zadań od 6 do 24 dla DPA, TSM i GA

Wykres 1. Porównanie średniego czasu wykonania dla ilości zadań od 6 do 24 dla DPA, TSM i GA

Wykres 2. Porównanie średniego czasu wykonania dla ilości zadań od 6 do 24 dla TSM i GA (dla lepszej widoczności)

1. Porównanie znalezionych wyników do wyników optymalnych dla GA i TSM

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Instancja | Wynik optymalny | TSM (Tabu Search Method) | | | GA (Genetic Algorithm) | | |
| Średni wynik obliczony | Błąd względny | Śr.czas wyk. [ms] | Średni wynik obliczony | Błąd względny | Śr.czas wyk. [ms] |
| wt40a | 913 | 930 | 0,0186 | 13,9485 | 1169 | 0,2804 | 1469,27 |
| wt40b | 1225 | 1230 | 0,0041 | 13,5250 | 1374 | 0,1216 | 1449,09 |
| wt40c | 537 | 573 | 0,0670 | 13,6862 | 573 | 0,0670 | 1439,93 |
| wt50a | 2134 | 2167 | 0,0155 | 28,8084 | 2184 | 0,0234 | 3124,72 |
| wt50b | 1996 | 2011 | 0,0075 | 29,2038 | 2242 | 0,1232 | 3121,21 |
| wt50c | 2583 | 2691 | 0,0418 | 30,0161 | 2646 | 0,0244 | 3157,94 |
| wt100a | 5988 | 6294 | 0,0511 | 311,3190 | 6874 | 0,1480 | 35759,1 |
| wt100b | 6170 | 6182 | 0,0019 | 315,7750 | 6620 | 0,0729 | 35731,2 |
| wt100c | 4267 | 4350 | 0,0195 | 311,2910 | 4603 | 0,0787 | 36414,7 |

Tabela 2. Porównanie wyniku właściwego i średniego wyniku wygenerowanego

1. **Wnioski**

Na czas wykonywania algorytmu genetycznego wpływają praktycznie wszystkie parametry. Większa ilość osobników to więcej krzyżowań i dłuższe ich sortowanie. Szansa mutacji wpływa na czas wykonania tylko jeśli wylosuje, żeby faktycznie przeprowadzić mutację, trudno więc przewidzieć ze względu na tą losowość. Najbardziej jednak znaczącym parametrem jest ilość iteracji co jest zrozumiałe, ze względu na to, że jest to równe ilości generacji, wiec wszystkie powyżej określone operacje zostaną powtórzone właśnie tyle razy.

Na podstawie badań można też wywnioskować, że wielkość instancji nie wpływa tak bardzo na różnicę miedzy uzyskanymi wynikami a wynikiem najbardziej optymalnym. Znaczenie najprawdopodobniej mają tutaj osobniki wynikające z krzyżowań i mutacji. Są to wyniki losowych działań, nie da się więc przewidzieć tak naprawdę ile iteracji potrzeba algorytmowi aby osiągnąć konkretne wyniki.

Podsumowywując wszystkie algorytmy:

DPA jest algorytmem służącym do znalezienia optymalnego i najbardziej dokładnego wyniku ze wszystkich tu opisywanych. Wykorzystuje dane zebrane z poprzednich iteracji algorytmu. Jest to jednak okupione dużymi wymaganiami pamięciowymi i czasowymi.

TSM natomiast stosuje metody przeszukiwania lokalnego aby dotrzeć do najlepszego lokalnie rozwiązania i w ostateczności dążyć do rozwiązania globalnie optymalnego. Jest mało złożony pamięciowo jak i czasowo, jest jednak mocno zależny od punktu startowego i parametrów.

GA jest algorytmem ewolucyjnym. Mocno skupiony na losowości dąży do ciągłego polepszania wyniku. W porównaniu z innymi algorytmami (metodami) pozostawia dużo możliwości na ciągłe jego ulepszanie. Nie jest złożony pamięciowo ani czasowo tak jak DPA, jest jednak pod tym względem gorszy niż TSM. Znajduje też rozwiązania od niego mniej optymalne.

1. **Literatura/źródła**

[**http://zio.iiar.pwr.wroc.pl/pea/w9\_ga\_tsp.pdf**](http://zio.iiar.pwr.wroc.pl/pea/w9_ga_tsp.pdf)

[**https://www.researchgate.net/publication/4011607\_A\_genetic\_algorithm\_for\_the\_single\_machine\_total\_weighted\_tardiness\_problem**](https://www.researchgate.net/publication/4011607_A_genetic_algorithm_for_the_single_machine_total_weighted_tardiness_problem)

[**http://www.obitko.com/tutorials/genetic-algorithms/operators.php**](http://www.obitko.com/tutorials/genetic-algorithms/operators.php)