**Optymalizacja Kombinatoryczna – laboratoria**

Wykonali:

Marcin Staszak 127241 I5

Piotr Furmankiewicz 127282 I5

**Temat projektu:**

Problem 1

Flowshop, liczba maszyn m=2, liczba zadań n,

operacje niewznawialne,

dla pierwszej i drugiej maszyny po k okresów przestoju (na każdą maszynę),

o losowym czasie rozpoczęcia i trwania (określonym przez generator instancji problemu), okresy te nie mogą się na siebie nakładać (tj. czas przestoju

na I maszynie nie może nakładać się z czasem dowolnego przestoju

na II maszynie), k >= n/10,

*minimalizacja sumy czasów zakończenia wszystkich operacji*

**Obsługa programu:**

Ścieżka do gotowego projektu w Visual Studio:

OK/projekt\_ok/projekt\_ok.sln

Rozwiązane instancje:

OK/Solutions/

Wygenerowane instancje:

OK/Instances/

**Opis działania algorytmu:**

**Generator instancji** losuje czas trwania operacji pierwszej oraz drugiej dla danego zadania z przedziału [1,ustalona wartość], „ready time” dla pierwszej operacji z przedziału [0,ustalona wartość] (dla drugiej nie, bo „ready time” dla operacji drugiej to czas zakończenia operacji pierwszej). Następnie losuje maintenance’y dla pierwszej i drugiej maszyny w taki sposób, że się na siebie nie nakładają. Czas i liczba maintenance’ów jest ustalona. Powyższe operacje generator wykonuje tyle razy, ile jest zadań.

**Generator rozwiązań losowych** dla każdej z maszyn wstawia najpierw maintenance’y, a następnie w pierwsze wolne miejsce w czasie każdą operację. Na końcu obie maszyny są sortowane wg czasu startu operacji.

Zaimplementowany algorytm składa się kolejno z mutacji, krzyżowania i selekcji.

**Mutacja**

**Krzyżowanie** losuje dwa już istniejące rozwiązania, z każdego z nich bierze połowę operacji i wstawia do nowych rozwiązań, a następnie drugą połowę uzupełnia kolejno brakującymi operacjami z przeciwnego rozwiązania. Podczas jednej iteracji krzyżowanie generuje 100 nowych rozwiązań.

**Selekcja** składa się z dwóch części turnieju oraz ruletki.

Turniej polega na wybieraniu najlepszych rozwiązań, a ruletka na losowaniu rozwiązań, które mają pozostać.

Liczba rozwiązań wybieranych przez turniej początkowo jest równa 30% i jest zwiększana z każdą sekundą o 10%. W sumie po każdej iteracji pozostawianych jest 100 rozwiązań.

Wykres nr 1.

Opis wykresu nr 1:

Oś Y - czas końcowy wygenerowany przez metaheurystykę.

Oś X - liczba zadań.

Wnioski:

Test został przeprowadzony dla stałej liczby maintenance’ów oraz dla stałego czasu ich trwania. Czasy trwania zadań losowane były z przedziału 1-20 jednostek czasu, a parametr „ready time” z przedziału 0-6. Jest to zależność oczywista, lecz przedstawiona ona jest po to, by dowieść, że metaheurystyka działa dobrze także w tym aspekcie.

Wykres nr 2.

Opis wykresu nr 2:

Oś Y - procent poprawy między sumą czasów zakończeń operacji uszeregowanych przez generator losowy a sumą czasów zakończeń operacji uszeregowanych przez metaheurystykę.

Oś X - liczba zadań.

Wnioski:

Wykres nr 2 został zrobiony dla takich samych warunków początkowych jak wykres nr 1. Wykres nr 2 przedstawia poprawę funkcji celu między rozwiązaniami wygenerowanymi przez generator losowy a rozwiązaniem końcowymi. Wniosek nasuwa się jeden, czym więcej zadań (dla tej samej liczby maintenance’ów), tym metaheurystyka działa gorzej. Im większa liczba zadań, tym gorsza funkcja celu, a gdy jednocześnie nie zwiększymy ilości maintenance’ów, metaheurystyka nie ma jak poprawić rozwiązania, bo jest mniej luk do wypełnienia.

Jeśli wraz ze wzrostem liczby zadań zwiększymy liczbę maintenance’ów wyniki wychodzą inne.

Tabela nr 1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Dane \ indeksy | 1 | 2 | 3 | 4 | 5 | 6 |
| Liczba maintenance'ów na każdej maszynie | 5 | 10 | 15 | 20 | 25 | 30 |
| Liczba zadań | 50 | 60 | 70 | 80 | 90 | 100 |

Wykres nr 3.

Opis wykresu nr 3:

Oś Y - procent poprawy między sumą czasów zakończeń operacji uszeregowanych przez generator losowy a sumą czasów zakończeń operacji uszeregowanych przez metaheurystykę.

Oś X - każdy indeks z tabeli nr 1 odpowiada instancji o danej liczbie zadań i maintenance’ów.

Wnioski:

Gdy wraz ze wzrostem liczby zadań zwiększymy liczbę maintenance’ów, to procent poprawy sumy czasów zakończeń operacji uszeregowanych przez metaheurystykę względem sumy czasów zakończeń operacji uszeregowanych przez generator losowy wzrasta prawie liniowo. Dzieje się tak dlatego, że powstaje więcej luk, które metaheurystyka może poprawić.

Tabela nr 2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 |
| Długość maintenance'ów | 30 | 25 | 20 | 15 | 10 | 5 |
| Liczba maintenance'ów | 5 | 10 | 15 | 20 | 25 | 30 |

Wykres nr 4.

Opis wykresu nr 4:

Oś Y - procent poprawy między sumą czasów zakończeń operacji uszeregowanych przez generator losowy a sumą czasów zakończeń operacji uszeregowanych przez metaheurystykę.

Oś X - każdy indeks z tabeli nr 2 odpowiada instancji o danej liczbie maintenance’ów i danym czasie trwania maintenance’ów.

Wnioski:

Wykres nr 4 miał na celu pokazać jak na działanie metaheurystyki wpłyną:

- wiele krótkich maintenance’ów,

- niewiele długich maintenance’ów.

Im więcej maintenance’ów o mniejszej długości, tym rozwiązanie końcowe jest lepsze w stosunku do początkowego.

Tabela nr 3.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Indeks | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Długość operacji | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |

Wykres nr 5.

Opis wykresu nr 5:

Oś Y - procent poprawy między sumą czasów zakończeń operacji uszeregowanych przez generator losowy a sumą czasów zakończeń operacji uszeregowanych przez metaheurystykę.

Oś X - każdy indeks z tabeli nr 3 odpowiada instancji o danych czasach trwania pojedynczych operacji. Jeśli w wierszu tabeli nr 3 „długość operacji” jest równa 20, oznacza to, że operacja może mieć długość z przedziału 1-20;

Wnioski:

Wykres może wydawać się niezbyt poukładany, lecz gdy popatrzy się na różnice między wartościami na osi Y, to te różnice okazują się być niewielkie. Procent poprawy rozwiązania waha się od 8% do 12%, więc jest to odchylenie do przyjęcia. Wykres przedstawia zależność w miarę stałą.

**Test ilości mutacji w pojedynczej iteracji.**

Wykres nr 6

Opis wykresu nr 6

W naszym programie wraz z każdą sekundą wykonywania zmniejszamy liczbę mutacji o 10, aż do 0.

Oś X - maksymalna(początkowa) liczba mutacji.

Oś Y – ostateczna wartość funkcji celu.

Wnioski:

Jak można zauważyć na wykresie w naszym programie optymalna liczba mutacji to 30, jednak jeśli nie wykonamy żadnych mutacji wartość funkcji celu będzie niewiele wyższa.

**Test czasu trwania algorytmu.**

Wykres nr 7

Opis wykresu nr 7

Oś X – czas trwania programu podany w sekundach

Oś Y – ostateczna wartość funkcji celu

Wnioski:

Funkcja celu osiąga najlepszą wartość przy czasie trwania programu równym 10.

Można zauważyć, że im dłuższy czas działania tym lepsze są wyniki generowane przez algorytm, jednak największą poprawę obserwujemy w sekundach 1-4, potem najprawdopodobniej program wpada w optimum lokalne.

**Test proporcji podczas selekcji.**

Jak to zostało opisane na początku sprawozdania selekcja wykorzystuje turniej oraz ruletkę w zmiennych proporcjach. Wartość procentowa rozwiązań wybieranych przez turniej jest co sekundę zwiększana o 10 aż do 100.

Liczby podane na osi x wykresu są to wartości początkowe.

Wartość funkcji celu to średnia z 10 prób dla każdej

z proporcji.

Wykres nr 8

Opis wykresu nr 8

Jak to zostało opisane na początku sprawozdania selekcja wykorzystuje turniej oraz ruletkę w zmiennych proporcjach. Wartość procentowa rozwiązań wybieranych przez turniej jest co sekundę zwiększana o 10 aż do 100.

Oś X – początkowe wartości proporcji podane w procentach

Oś Y – Ostateczna wartość funkcji celu

Wnioski:

Można zauważyć , że proporcja początkowa 50 % jest optymalna. Jeśli zwiększymy ją lub zmniejszymy wartości funkcji celu wzrosną.