Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Elektroniki, Automatyki, Informatyki i Inżynierii Biomedycznej
Informatyka – 2 rok, semestr 4
2013/2014

Badania operacyjne i teoria złożoności

Projekt zaliczeniowy

Wojciech Kryściński
Aleksander Wójcik
Filip Pasternak

Problem harmonogramowania

SPIS TREŚCI

Opis problemu

Problem harmonogramowania

Problem harmonogramowania jest często obecny w codziennym życiu. To istotny element procesów takich jak zarządzanie zapasami, projektami, planowanie produkcji, układanie planów zajęć i tym podobne.

Niezależnie od konkretnego celu, procesy decyzyjne można określić następująco: dany jest zbiór zadań, zbiór procesorów oraz ograniczone zasoby przeznaczone do ich realizacji. Szukamy przydziału zadań do procesorów, kolejności wykonywania oraz rozdział ograniczonych zasobów spełniających zadane w konkretnym problemie warunki tak, aby uzyskać optymalną wartość przyjętego kryterium jakości. Przykładem procesora może być stanowisko, środek transportu, maszyna czy sala wykładowa.

Podstawowym problemem jest kwestia spełnienia ograniczeń. Uznajmy, iż istnieje pewien zbiór zmiennych (decyzyjnych), z których każda może przyjmować pewne wartości. Rozwiązaniem problemu jest przyporządkowanie powyższym zmiennym takich wartości, aby wszystkie ograniczenia zostały spełnione jednocześnie. Zdarzają się także sytuacje, gdzie nie jesteśmy w stanie uzyskać takiego wyniku. Wówczas można zastosować priorytetowanie ograniczeń – niektóre warunki koniecznie muszą zostać spełnione, gdy inne są sprawą drugorzędną lub zupełnie opcjonalną – i wyznaczyć pewien współczynnik jakości uzyskanych wyników.

„JobShop” jako problem harmonogramowania

Nasz projekt to implementacja systemu obsługi typu gniazdowego - „Job Shop”.

Dany jest zestaw maszyn (Machine), które mogą być użyte do wykonania pewnych czynności. Istnieje również zbiór prac (Job), do których wykonania dążymy. Każda z prac składa się z sekwencji operacji (Task) będących kolejnymi krokami prowadzącymi do zakończenia pracy. Do przeprowadzenia operacji użyta zostaje konkretna maszyna.

W tym przypadku problem jest formułowany przy użyciu ograniczeń kolejnościowych oraz zasobowych. Zmiennymi decyzyjnymi są czasy rozpoczęcia operacji poszczególnych prac.

Przy pomocy programu wyszukujemy taki rozkład użycia dostępnych maszyn, aby wszystkie zadane do wykonania prace zostały wykonane w jak najkrótszym czasie przy zachowaniu odpowiedniej kolejności przeprowadzanych operacji w obrębie danej pracy oraz przy założeniu, że w konkretnej chwili czasu jedna maszyna może być użyta do maksymalnie jednego zadania.

Przykładowe pytania na jakie możemy uzyskać odpowiedź:
Ile czasu potrzeba do wykonania wszystkich podanych zadań przy obecnych zasobach?

W jakiej kolejności należy używać dostępnych zasobów, aby uzyskać optymalny wynik?

Budowa i znaczenie chromosomu

W teorii algorytmów genetycznych każdy chromosom reprezentuje jedno poprawne (w wersji algorytmu zaimplementowanej w projekcie) rozwiązanie problemu.

Najważniejszym elementem każdego chromosomu jest jego genotyp, który to właśnie rozwiązanie przechowuje. W naszej implementacji zdecydowaliśmy się reprezentować go jako vector zmiennych typu integer, które symbolizują kolejność brania pod uwagę poszczególnych prac przy reprezentacji rozwiązania. Dzięki takiemu rozwiązaniu krzyżowanie osobników jest stosunkowo proste i nie wymaga żadnych konwersji (jeden wektor przechowuje pełne rozwiązanie).

Miarą przystosowania danego osobnika (reprezentowanego przez chromosom), do danej populacji, jest wartość tzw. funkcji przystosowania. Dla problemu harmonogramowania funkcja ta zwraca czas jaki upłynie między rozpoczęciem a zakończeniem wszystkich prac. Naturalnie im owy czas jest mniejszy tym lepiej. Krzyżowaniu więc podlegają te osobniki których w tym problemie wartość funkcji przystowania będzie mniejsza. W naszej implementacji zmienna fitness oraz funkcja wyliczająca ową zmienną są zdefiniowane jako elementy składowe klasy chromosom.

Ostatnimi elementami składowymi chromosomu zastosowanymi w projekcie są wskaźnik do bazy danych, dzięki któremu chromosom może wykonywać szereg funkcji sprawdzających poprawność czy też realizować zapisywanie najlepszych rezulatatów do owej bazy.

Operatory genetyczne

Operatory mutacji:
Point Swap Mutation
Z genotypu wybierane sądwa geny które następnie sązamieniane miejscami.

Przykład działania:
Genotyp przed mutacja: 1 **1** 2 3 4 **2** 4 3
Genotyp po mutacji: 1 **2** 2 3 4 **1** 4 3

Segment Swap Mutation
Z genotypu wybierane sądwa ciągi genów które następnie zamieniane miejscami.

Przykład działania:
Genotyp przed mutacja: **1 1** 2 3 4 **2 4 3**
Genotyp po mutacji: **2 4 3** 2 3 4 **1 1**

Reverse Mutation
Kolejnośćgenów w genotypie jest odwracana.

Przykład działania:
Genotyp przed mutacja: **1 1 2 3 4 2 4 3**
Genotyp po mutacji: **3 4 2 4 3 2 1 1**

Inversion Mutation
Z genotypu wybierany jest ciąg genów o losowej długości, którego kolejnośćjest odwracana.

Przykład działania:
Genotyp przed mutacja: 1 1 2**3 4 2 4** 3
Genotyp po mutacji: 1 1 2**4 2 4 3**3

Displacement Mutation
Z genotypu wybierany jest ciąg genów o losowej długości, który przenoszony jest w inne, losowy wybrane miejsce, genotypu.

Przykład działania:
Genotyp przed mutacja: 1 **1 2 3** 4 2 4 3
Genotyp po mutacji: 1 4 2 4 **1 2 3**3

Operatory krzyżowania:
One Point
Klasyczne krzyżowanie jednopunktowe. W genotypie wybierany jest punkt od którego następuje wymiana genotypów pomiędzy chromosomami.

Rodzic A: 1 1 2 **|3 4 2 4 3**
Rodzic B: 1 2 1 **| 4 2 3 3 4**
Dziecko A: 1 1 2 **|4 2 3 3 4**
Dziecko B 1 2 1 **|3 4 2 4 3**

Two Point
Klasyczne krzyżowanie dwupunktowe. W genotypie wybierane sądwa punkty. Genotyp pomiędzy punktami wyboru jest brany od jednego rodzica, pozostała częśćgenotypu brana jest od drugiego rodzica.

Przykład działania:
Rodzic A: **1 1 2|** 3 4 2 **| 4 3**
Rodzic B: **1 2 1 |** 4 2 3 **| 3 4**
Dziecko A: **1 2 1|** 3 4 2 **| 3 4**
Dziecko B **1 1 2|** 4 2 3 **| 4 3**

Uniform
Przy każdym genie w genotypie następuje losowanie z którego rodzica ma zostaćprzepisany gen. Zazwyczaj rodzice mająrówne szanse bycia ‘dawcami’.

Przykład działania:
Rodzic A: 1 1 2 **3** 4 **24** 3
Rodzic B: **141** 4 **2** 3 2 **3**
Dziecko A: **1 4 1** 3 **2** 2 4 **3**
Dziecko B 1 1 2 4 4 3 2 3

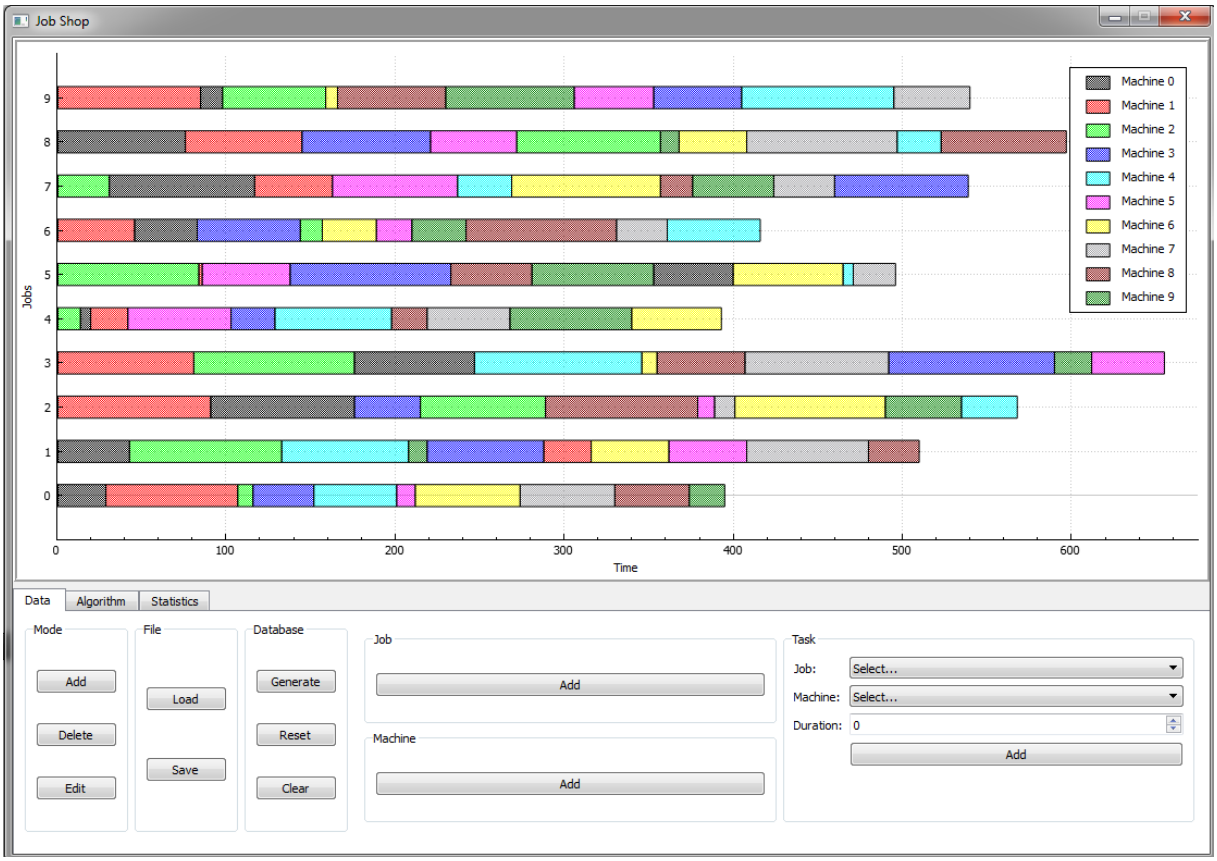
Order
Działanie krzyżowania jest analogiczne do krzyżowania dwupunktowego, różnica polega w sposobie dostawiania pozostałej części genotypu od drugiego rodzica.
Przykład działania:
Rodzic A: **1 1 2 |** 3 4 2 **| 4 3**
Rodzic B: **1 4 1 |** 4 2 3 **| 2 3**
Dziecko A: **1 2 1|** 3 4 2 **| 3 4**
Dziecko B **1 1 2|** 4 2 3 **| 4 3**

Operatory selekcji:
Rank Selection
Selekcja rankingowa opiera sięna jakości dostosowania osobników na tle populacji. Osobniki najlepiej przystosowane mająnajwiększąszansęwyboru.

Tournament Selection
Selekcja turniejowa polega na wybranej określonej ilości losowych osobników z całej populacji a następnie wyboru najlepszego spośród z nich z danym prawdopodobieństwem.

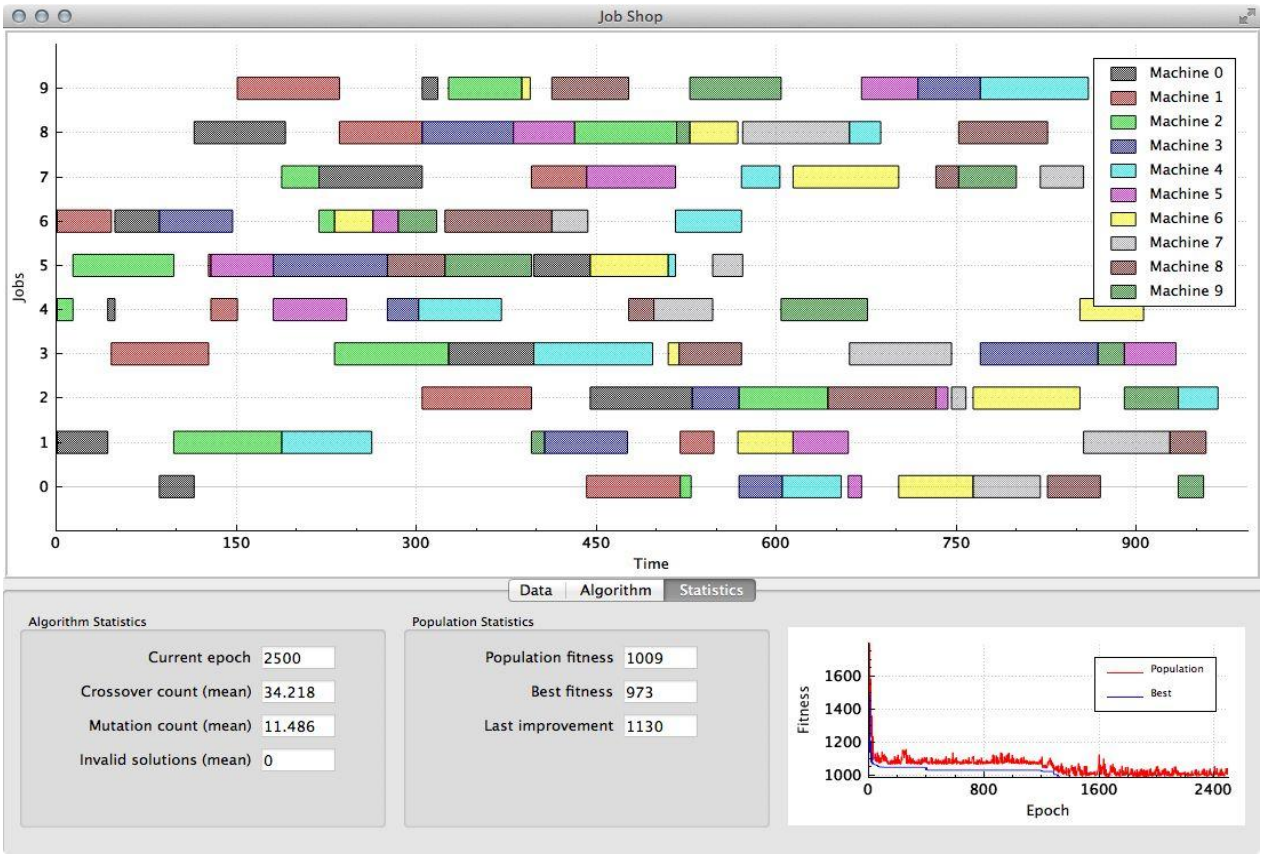
Analiza przykładowych danych wynikowych

Po wczytaniu przykładowego pliku w centralnej części okna ukazuje się wykres pokazujący dane wejściowe problemu (liczba prac oraz maszyn, kolejność wykonywania zadań). Ważne jest to, iż nie jest to jeszcze żadne rozwiązanie problemu (wiele bloczków reprezentujących te same maszyny zajmuje ten sam przedział czasowy). Wykres ten jest rysowany jedynie w celach informacyjnych, aby zwizualizować sobie daną bazę danych.



Po ustawieniu ineresujących nas parametrów algorytmu w zakładce „Algorithm” i kliknięciu przycisku „Run” uruchamia się sam algorytm. Centralne okno od tego momentu zaczyna reprezentować najlepsze dotychczas znalezione rozwiązanie.

Najważniejszym elementem zakładki „Statistics” jest pole „Best fitness”. To właśnie pole wyświetla niejako wynik algorytmu; najkrótszy znaleziony czas w jakim wykonają się wszystkie prace.



Każdy bloczek na centralnym wykresie obrazuje jedno zadanie. Informacje o konkretnym zadaniu można uzyskać klikając na interesujący na box. Wyświetli się okienko informujące o czasie rozpoczęcia oraz trwania danego użycia maszyny.

Dla wczytanego problemu znane jest najlepsze rozwiązanie, w którym czas wykonania wszystkich prac wynosi 824. Modyfikując ustawienia algorytmu (zwiększając liczbę epok, liczbę osobników w populacji itp.) udało się nam zmniejszyć owy parametr do ok. 900(„fitness" rozwiązania przedstawionego na screenie wynosi 973). Pytanie czy jest dobry wynik pozostawiamy otwartym do interpretacji. Należy jedynie dodać, że dla generowanego losowo genotypu, jego przystosowanie oscyluje w okolicach 2000. Osczędzamy więc ok. połowy potrzebnego czasu.