Aukcje kombinatoryczne

Piotr Rzepecki, Krzysztof Zielonka

4 lutego 2013

1 Wstęp

1.1 Opis problemu

W aukcjach kombinatorycznych (ang. combinatorial auction) przedmiotem handlu jest wiele towarów. Uczestnicy mogą składać oferty na zbiory towarów i te oferty są niepodzielne, tzn. muszą być przyjęte w całości lub w całości odrzucone. Problem wyznaczania zbioru ofert przyjętych maksymalizujących przychód w takiej aukcji jest w ogólnym przypadku NP trudnym problemem kombinatorycznym.

1.2 Dane i rozwiązania

Dane to liczba towarów n i m ofert, gdzie każda oferta to lista towarów i proponowana za nią cena. Oferty te są niepodzielne, a każdy przedmiot może zostać kupiony tylko raz.

Rozwiązaniem nazywamy zbiór ofert, w którym żadne dwie oferty nie zawierają tego samego przedmiotu.

1.3 Funkcja celu

Funkcją celu jest suma wartości ze zbioru A wybrancyh ofert (przy czym zbiór ten spełnia wymagania zadania i oferty są niesprzeczne).

$$f(A) = \sum_{a \in A} cena(a) \tag{1}$$

1.4 Reprezentacja danych

Uwagi:

- naturalną reprezentacją byłby zbiór identyfikatorów ofert,
- ta reprezentacja nie pozwala jednak na efektywne stosowanie operatorów genetycznych

Alternatywy:

- wektor binarny
- permutacja

1.5 Dane testowe

Do generowania danych testów korzystamy z generatora CATS. Jest on najbardziej popularnym narzędziem dla tego problemu i jak twierdzi autor generuje dane zbliżone dla realnych problemów tego typu.

2 Rozwiązanie przy użycia wektora binarnego

2.1 Reprezentacja danych

Reprezetancja za pomoca wektora binarnego gdzie jedynke interpretujemy jako kondydature oferty do zaakceptowania, a zero za brak oferty w zbiorze ofert zaakceptowanych. Jeżeli oferta posiada jedynke w wektorze czyli jest kandydatem do zaakceptowania nie oznacza jeszcze, że zostanie dodana do zbioru ofert zaakceptowanych.

Dzięki takiej reprezentacji możemy zastosować algorytm PBIL do rozwiązania tego problemu.

2.2 Funkcja celu

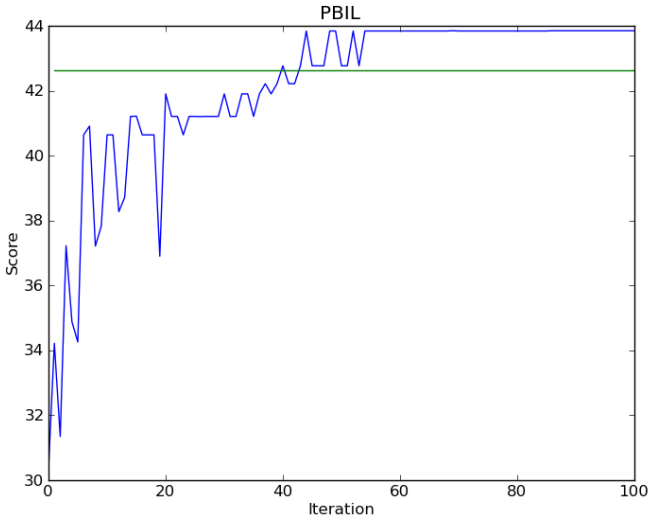
Oferty oznaczone jedynka rozumiemy jako oferty zaakceptowany. Dopuszczamy jednak możiwość występownia dwóch ofert zaakceptowanych o nierozłoącznym zbiorze towarów, ale wtedy preferujemy tą o mniejszym indeksie

2.3 Rozwiązanie

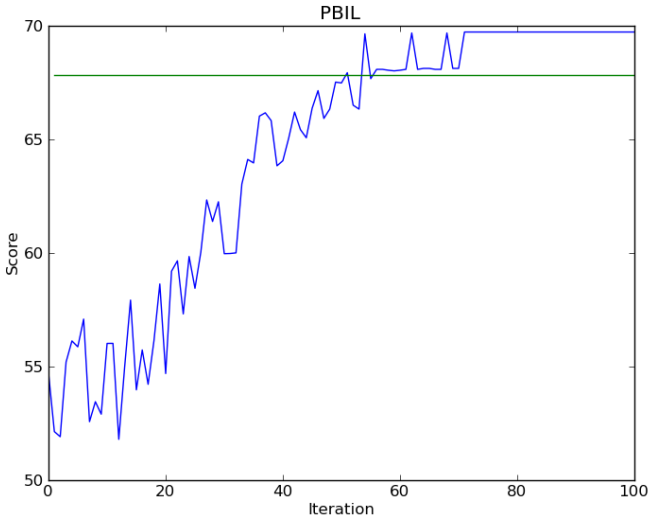
Do rozwiązania tego problemu użyliśmy algorymu PBIL. Współczyniki uczenia, prawdopodobieństwa mutacji i zaburzenia podczas mutacji ustawiliśmy kolejno na 0.2, $\frac{1}{\text{liczba ofert}}$ i 0.1. Liczbe iteracji ustawialiśmy na 100, a populacje na 20.

2.4 Wyniki

Wyniki zaprezentowane są na wykresach poniżej (1, 2). Niebieską linią oznaczyliśmy najepszego osobnika dla algorytmu PBIL w danej iteracji. Zieloną najlepszy wynik algorytm losowego, który wybierał z takiej samej puli losowych osobników ile łącznie przegląda PBIL.



Rysunek 1: Wykres dla problemu 'matching' z 20 ofertami i 100 towarami.



Rysunek 2: Wykres dla problemu 'matching' z 40 ofertami i 100 towarmi.

3 Rozwiązanie przy użyciu permutacji

3.1 Reprezentacja rozwiązania

Rozwiązaniem problemu jest permutacja długości liczby ofert. Za oferty przyjęte jako zaakceptowane wybieramy zgodnie z kolejność występowania w permutacji

z pominięciem tych, których towary pokrywały się z towarami jakiejs oferty wcześniej występującej.

3.2 Funkcja celu

Zgodnie z reprezentacją sumujemy te ceny tych ofert, które zostały zaakceptowane. Przeglądamy permutacje od lewej do prawej i wybieramy te oferty, których towary nie zostały wykupione przez wcześniejsze oferty.

3.3 Rozwiązanie

Do rozwiązania tego problemu użyliśmy algorytmu algorytmu SGA. Jego parametry najcześciej na 200 iteracji i populacje o rozmiarze 40. Podczas każdej iteracji, nie wymienialiśmy całej populacje, a jedynie 10 osobników.

3.3.1 Krzyżowanie

Do krzyżowania użyliśmy lekko zmodyfikowanego algorytmu PMX. Założyliśmy, że funkcja celu zależy bardziej od elementów o mniejszym indeksie w permutacji niż większym. Elementy o mniejszy indeksie są bardziej preferowane co wynika z funkcji celu, a oferty o dalszych indeksach mogą być często wogle nie brane do rozwiazania. Dlatego staramy się, aby wymieniamy przez operator PMX środkowy segment zaczynał się cześciej w niskich indeksach.

Wprowadzona modyfikacja polega na wyborze punktów a i b będących przedziałem wymienianego środkowego segmentu. W pierwotnej wersji były losowane dwie liczby x, y i na ich podstawie wyliczane punkty a, b.

$$a = min(x, y) \cdot n \ \wedge \ b = max(x, y) \cdot n \ \text{ gdzie n to długość permutacji} \qquad (2)$$

W zmodyfikowanej wersji punty były wyliczane w poniższy sposób.

$$a = min(x, y)^2 \cdot n \ \wedge \ b = max(x, y) \cdot n \qquad (3)$$

3.3.2 Mutacja

Wybrana przez nas mutacja jest dość prosta, ale okazała się znacząco poprawiać wyniki. Polega na cyklicznym przesunięciu całej permutacji o jeden element w lewo. W ten sposób oferta, która była pierwsz w permutacji staje się ostatnio. Z zasady działania funkcji celu wielmu, że pierwsza oferta jest zawsze wybierana, a ostatnio dość rzadko dzięki czemu ta mutacja dość znacząco zmieniała osobniki.

3.3.3 Wymiana

Wymiana elemntów polega na zastąpieniu najgorszych osobników stałą liczbą, ustawianą jako parametr, zmutowanych i skrzyżowancyh osobników. Dzięku odpowiednim doborze tej liczby populacja zawsze pamiętała stare najlepsze osobniki przez pewną liczba iteracji co wpływało na poprawę znajdowanych wyników.

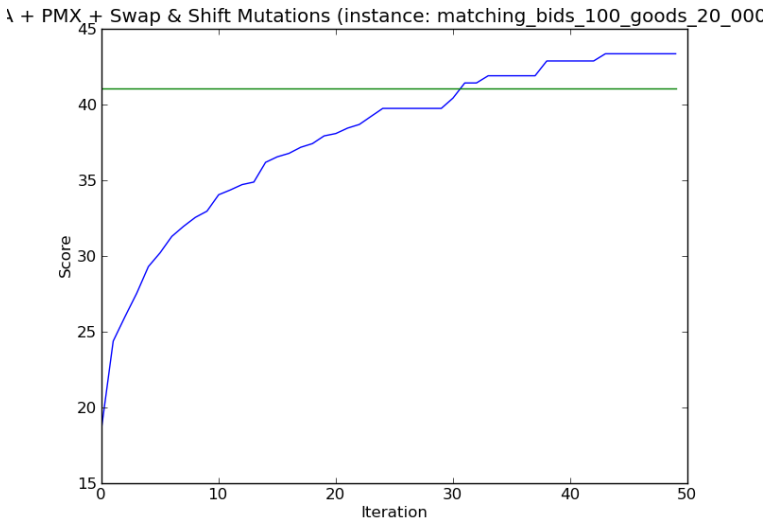
3.3.4 Warunek stopu

Na początku za warunek stopu przyjeliśmy unikalność wszystkich elementów. Gdy wartości funkcji celu dla wszystkich elementół w populacji były takie same to przerywaliśmy algorytm. Takie podejście okazało się jednak błędne, gdyż pomimo iż wartości funkcji celu były takie same osobniki były znacząco różne w sensie permutacji i pozwolenie im na dalsze iteracje poprawiło trochę wyniki.

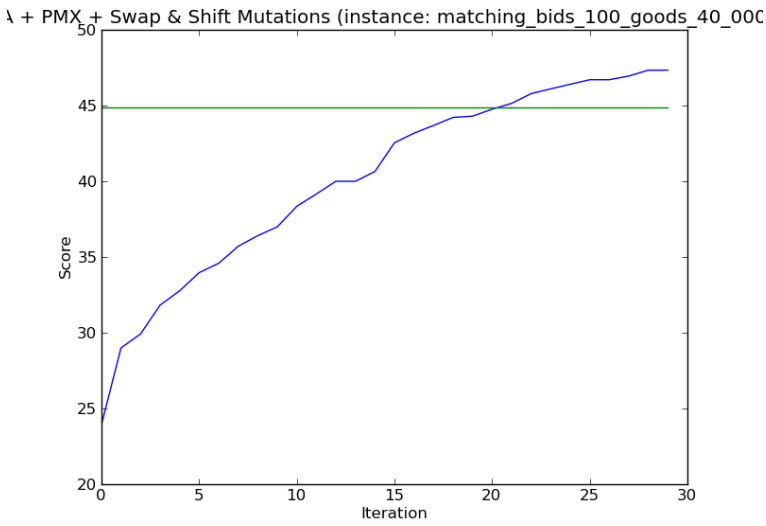
3.4 Wyniki

Wyniki przestawione są na wykresach (3, 4, 5, 6). Na niebiesko zaznaczono wynik dla algorytmu SGA dla kolejnych iteracji. Na zielono zaznaczono najlepszy wynik algorytmu losowego, który wybierał wynik z takiej samej puli losowych osobników ile łącznie przegląda SGA.

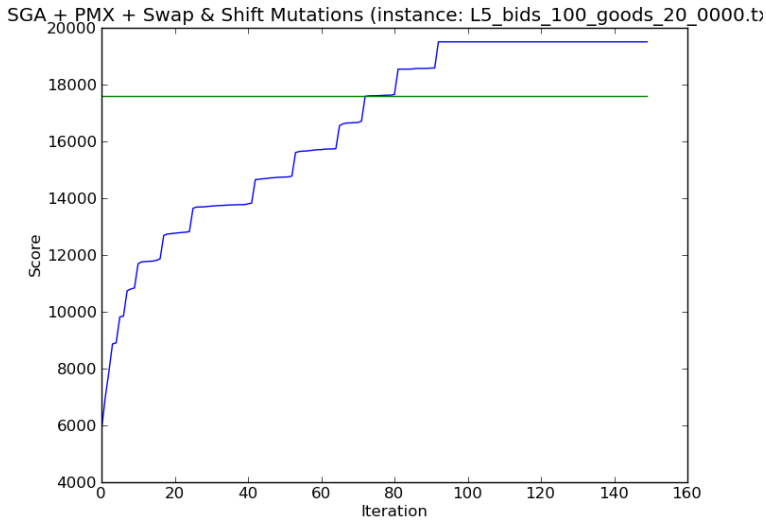
Na wykresach 3, 4 i 5 algortym kończył swoje działanie gdy wszsytkie osobniki w populacji były unikalne pod względem wartości funkcji celu. Na wykresie 6 przestawiona jest wersja algorytmu, która kończyła poszukiwania po ustalonej liczbie iteracji. Podczas algorytmu próbowaliśmy również podmieniać niektóre stare osobniki nowymi losowymi gdy wszystkie poprzednie były już unikalne. Jak widać takie podejście okazało się być lepsze od poprzedniego gdyż algorytm po pewnym czasie działania znajdował troche lepsze rozwiązanie.



Rysunek 3: Wykres dla problemu 'matching' o parametrach 100 ofert i 20 towarów.



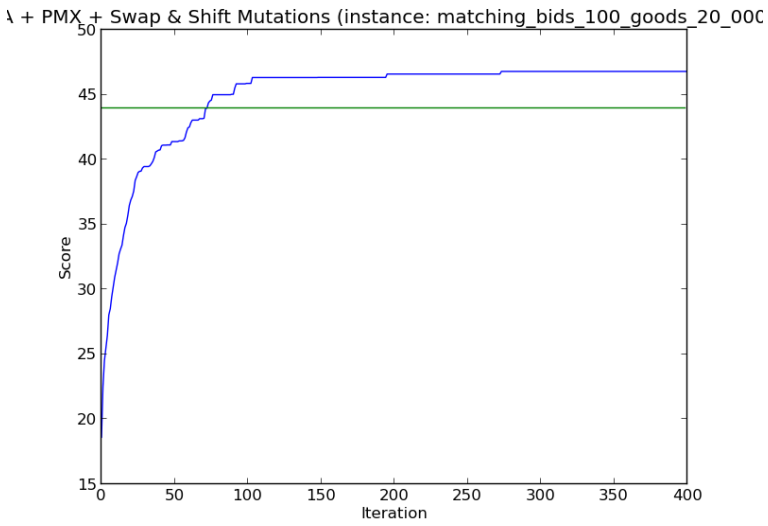
Rysunek 4: Wykres dla problemu 'matching' o parametrach 100 ofert i 40 towarów.



Rysunek 5: Wykres dla problemu 'L5' o parametrach 100 ofert i 20 towarów.

4 Porównanie obu algorytmów

Przeprowadzone testy wskazują, że zdecydowanie lepszy jest algorytm SGA. W tabeli poniżej (1) przestawione, są wyniki najelpszych rezulatów dla trzech



Rysunek 6: Wykres dla problemu 'matching' o parametrach 100 ofert i 20 towarów.

algorytmów. Algorytmy uruchamialiśmy wielektornie i wyniki przedstawione w tablece oddają najczęściej występującą tendencje. Zdarzało się, że algorytm losowy był lepszy od algorutmy PBIL, ale SGA prawie zawsze wyprzedzał oba te algorytmy.

Problem	Algorytm losowy	PBIL	SGA
matching, 100 ofert, 20 towarów	43.683	43.848	46.793
matching, 100 ofert, 40 towarów	65.543	67.188	72.785

Tablica 1: Porównanie algorytmów

5 Podsumowanie

Przeprowadzone testy wyraźnie pokazują, że algorytm SGA jest znacznie lepszy od algorytmów PBIL i algorytmu losowego. Zdarzało się, że algorytm losowy dawał prawie tak dobre wyniki jak SGA lub równe ale w większości przypadków był od niego gorszy. Algorytm PBIL okazał się być troche lepszy od algorutmy losowego, ale zdarzały się sytuacje, że oba były równe, a nawet algorytm losowy był lepszy. Zaóważyliśmy, że trudność danych testowych dla algorytmu losowego, zależa od stosunku towarów do ofert. Algorytm SGA i PBIL okazały się być znacznie od niego lepsze gdy liczba ofert znacznie przewyższa liczbę towarów.

Wydaje nam się, że najtrudniejszymelementm w tych algorytmach to dobranie odpowiendiej reprezentacji danych. Reprezentacje które wybraliśmy mają tą wade, że niektóre zbiory zaakceptowanych ofert mogą być nierównomiernie reprezentowane przez osobników. W efekcie algorytm ma problemy ze znalezieniem optymalnego zbioru ofert, który może być reprezentowany przez bardzo małą liczbe osobników.