Studia i Materiały Informatyki Stosowanej, Tom 2, Nr 2, 2010
str. 47-50

ALGORYTM PSZCZELI W OPTYMALIZACJI MODELU PRZEPŁYWOWEGO SZEREGOWANIA ZADAŃ

Wiesław Popielarski

Akademia Górniczo-Hutnicza im. St. Staszica
Doktorant III roku
Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki
Al. Mickiewicza 30, 30-059 Kraków
e-mail: wieslaw.popielarski@sabre.com

Streszczenie: *Problem optymalizacji przy ograniczonych zasobach jest jednym z podstawowych tematów w informatyce. Algorytm pszczeli z szerszej grupy algorytmów stadnych, wynaleziony i przedstawiony w połowie ostatniej dekady, wydaje się być obiecującym narzędziem w optymalizacji kombinatorycznej. Artykuł przedstawia wyniki badań nad algorytmem w optymalizacji modelu przepływowego szeregowania zadań i zakreśla dalsze ich obszary.*

Słowa kluczowe: *Algorytm pszczeli, model przepływowy, szeregowanie zadań*

Bees algorithm in optimization of task scheduling for flow shop model

Abstarct: *Problem of optimization with limited resources is fundamental one in computer sciences. The bees algorithm from wider group of swarm algorithms, invented and implemented about 2005, seems to be a good candidate for next useful tool for combinatorial optimization. Article presents the results of the bees algorithm research in flow shop model of task scheduling and outlines areas of further exploration.*
Keywords: *Bees Algorithm, Flow Shop, Task Scheduling*

1. WPROWADZENIE

Dla wielu problemów algorytmicznych, które generalnie zaliczane są do klasy NP, nie spodziewamy się znaleźć rozwiązań, które dawałyby poprawne odpowiedzi w czasie wielomianowym dla dowolnego zbioru wejściowego. Dobrym przykładem mocnego algorytmu, który ma taką własność, jest metoda sympleks programowania liniowego. Z reguły jest on w stanie rozwiązać problemy w czasie wielomianowym jednakże możemy skonstruować takie zbiory danych, dla których to narzędzie będzie potrzebować wykładniczej ilości kroków, by podać odpowiedź. Ze względu na to, że programowanie liniowe jest algorytmem deterministycznym, odpowiedź zawsze będzie optymalna (pod warunkiem, że rozwiązanie optymalne istnieje, patrz „Podstawowe twierdzenie programowania liniowego" w [6]).

Zanim przejdziemy do głównego tematu artykułu, rozważmy przez chwilę problem zbioru danych wejściowych. Bardzo interesującą i sprzyjającą okolicznością jest fakt, że statystycznie zbiory niekorzystne, a więc nie dające się rozwiązać w czasie wielomianowym, występują stosunkowo rzadko. Oczywiście tylko wtedy, gdy rozwiązywany problem nie dotyczy albo nie ogranicza się do tych szczególnych danych. Co więcej, daje to możliwość zastosowania nie tylko algorytmów deterministycznych, ale także stochastycznych i chociaż w tym drugim przypadku znalezione rozwiązanie nie musi być rozwiązaniem optymalnym, to jest ono z reguły na tyle bliskie, że popełniany błąd jest kompensowany przez ilość wykonywanych kroków, dużo mniejszą jak w przypadku algorytmów deterministycznych.
Jedną z klas algorytmów stochastycznych są algorytmy stadne, do których zalicza się przedstawiany algorytm pszczeli. Jak można domniemywać, algorytm pszczeli jest procedurą przeszukiwania przestrzeni rozwiązań

naśladującą sposób poszukiwania i pozyskiwania pokarmu przez pszczoły. Pierwsze implementacje pojawiły się w drugiej połowie ostatniej dekady , a więc stosunkowo niedawno, i nadal są przedmiotem prac badawczych.

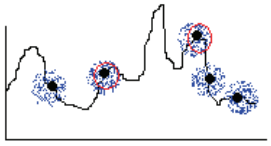
2. ALGORYTM PSZCZELI - STRUKTURA

Entomologia daje nam pewne wyobrażenie o strategii pozyskiwania pokarmu przez pszczoły. Początkowo część robotnic (zwiadowcy) poszukuje źródeł pokarmu wybierając w sposób losowy kierunek i obszar poszukiwań. Po znalezieniu potencjalnego źródła pożywienia, wraca do ula, by informacje o kierunku, odległości, obszarze i zasobności źródła pokarmu przekazać innym pszczołom w specyficznym rodzaju tańca (ang. waggle dance). Pozostałe pszczoły oceniają taniec zwiadowców i decydują się lub też nie na przyłączenie się do najlepiej tańczącego spośród nich. Generalnie im taniec jest dłuższy i bardziej ekspresyjny, tym więcej pszczół podąży za zwiadowcą. Zrobią to również te, które aktualnie eksploatują inne źródło, a które uznają, że nowo odnaleziony obszar rokuje na większą ilość pokarmu i efektywniejszy jego zbiór. W ten sposób kolonie pszczół rozwiązują problem maksymalizowania zbiorów przy ograniczonych zasobach.

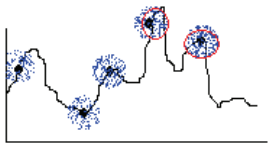
Algorytm pszczeli jest algorytmem iteracyjnym i z reguły jedynym kryterium stopu jest liczba iteracji podobnie, jak w przypadku innych algorytmów losowych. Wynika to przynajmniej z dwóch zasadniczych powodów. Po pierwsze, algorytm z reguły nie przeszukuje całej przestrzeni rozwiązań, a tylko jej część, a ponadto przeważnie ze względu na koszt, obliczane są jedynie pomocnicze wskaźniki jakości, które nie niosą pełnej informacji o zachowaniu się wskaźnika głównego. Również jak każdy algorytm losowy (genetyczny, symulowane wyżarzanie) algorytm pszczeli może być podatny na problem przedwczesnej zbieżności w zależności od przyjętej strategii przeszukiwania. Ogólne działanie algorytmu można opisać w następujących krokach:

1. Inicjalizacja populacji początkowej – wstępne określenie parametrów algorytmu takich jak wielkość populacji – *n*, liczba wyznaczonych rokujących obszarów w iteracji – *m*, liczba najlepszych rozwiązań w iteracji – *e* oraz inne; określenie wskaźnika jakości; przypisanie losowo wybranych obszarów przestrzeni rozwiązań poszczególnym zwiadowcom;
2. Ocena jakości znalezionych rozwiązań – wyznaczenie wskaźnika jakości dla każdej wartości znalezionej przez zwiadowcę; wskaźnik określa, czy zwiadowca odnalazł rokujący obszar; *m* najlepiej rokujących obszarów przechodzi do następnego kroku algorytmu;
3. Eksploracja najlepszych *m* rozwiązań – dla oznaczonych *e* spośród *m* rozwiązań przydzielamy większą ilość osobników do przeszukiwania otoczenia;
4. Wybór najlepszych *e* rozwiązań spośród *m* przeszukiwanych – rozwiązania te biorą udział w dalszej części algorytmu; tych *e* rozwiązań może się różnić od *e* początkowo znalezionych;
5. Eksploracja przestrzeni niezatrudnionymi osobnikami – rozlosowanie obszarów niezatrudnionym osobnikom;
6. Ocena jakości znalezionych rozwiązań - krok identyczny z krokiem 2
7. Iteracje – jeśli spełniono warunek stopu – zakończ; w przeciwnym wypadku skocz do kroku 3;

Na rysunku 1 pokazano przykład przeszukiwanego obszaru rozwiązań z zaznaczoną interpretacją wartości *m* oraz *e*.



a) punkty - reprezentują wylosowane wartości początkowe, chmury - przeszukiwane sąsiedztwo punktów, koła - najlepsze sąsiedztwa



b) trzy najgorsze obszary zostały zastąpione nowo wylosowanymi, zmieniły się też najlepsze sąsiedztwa - pozostało jedno z poprzedniej iteracji, drugie zostało znalezione w bieżącej

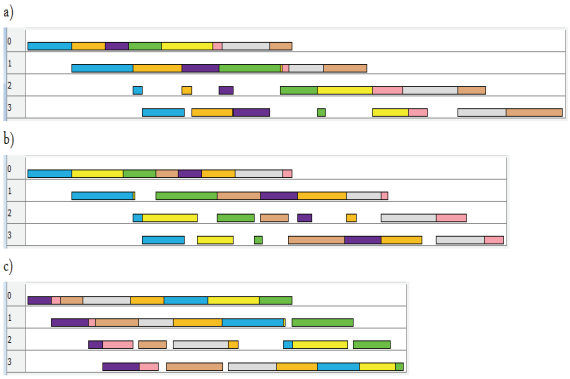
Rysunek. 1 Przykład przeszukiwanego obszaru dla *m=5* i *e=2*.

3. OPTYMALIZACJA MODELU PRZEPŁYWOWEGO

W modelu przepływowym *F* problemu szeregowania zadań (ang. flow shop) *n* zadań (ang. task) jest przetwarzanych na *m* maszynach. Każde zadanie *k* dzieli się na *m* prac (ang. job), z których każda wykonywana jest na innej maszynie. Stąd też każde z *n* zadań przetwarzane jest przez wszystkie maszyny dokładnie jeden raz. Istotnym założeniem dla modelu przepływowego jest kolejność maszyn, która jest taka sama dla wszystkich przetwarzanych zadań. W tym samym czasie maszyna może przetwarzać co najwyżej jedno zadanie oraz jedno zadanie może być przetwarzane przez co najwyżej jedną maszynę. Zadanie może być przetwarzane na następnej maszynie pod warunkiem, że jej aktualna praca została zakończona na maszynie bieżącej. W modelu dane są czasy t_{ij} wykonywania pracy zadania *j* na maszynie *i*. Optymalizacja polega na znalezieniu takiej permutacji zadań, dla której czas zakończenia ostatniego zadania na ostatniej maszynie jest najmniejszy. Czas ten oznaczany jest przez C_{max}. Równoważne jest to stwierdzeniu, że poszukiwana jest taka permutacja zadań, dla której sumaryczny czas bezczynności maszyn w oczekiwaniu na kolejne zadania jest minimalny.

Przykładowo zadanie *T1* wykonywane na czterech procesorach można zapisać jako: $T1=(j_0\ j_1\ j_2\ j_3)$, gdzie j_k oznacza pracę wykonywaną na procesorze *k*. Prace te wykonywane są zawsze w takiej samej kolejności, bez względu pozycję zadania *T1*. Stąd też czasy przetwarzania prac *j* mogłyby przyjmować wartości $t_{00}=a$, $t_{11}=a$, $t_{22}=a$, $t_{33}=a$, zakładając, że kolejność maszyn określona jest przez ciąg 0-1-2-3. Celem optymalizacji jest znalezienie takiej permutacji zadań *T1,…,Tn*, dla której całkowity czas przetwarzania będzie najkrótszy ([2] i [4]).

Rysunek 2 przedstawia przykłady trzech permutacji (uszeregowań) ośmiu zadań wykonywanych na czterech maszynach (stąd też każde zadanie składa się z czterech prac). Kolejność maszyn jest ustalona i określona w porządku 0-1-2-3 tak więc, każde zadanie najpierw musi być przetworzone przez maszynę 0, następnie 1, 2 i w końcu 3. Jeżeli przyjąć, że rysunek 2a) przedstawia uszeregowane zadania w kolejności 1-2-3-4-5-6-7-8, a rysunek 2c) przedstawia permutację optymalną, to kolejność optymalna wynosi 3-6-8-7-2-1-5-4.



Rysuek. 2 Przykładowe realizacje ośmiu zadań na czterech maszynach. Jednym kolorem oznaczono prace należące do tego samego zadania wykonywane na kolejnych maszynach (może zaskakiwać brak pracy zadania 5 oznaczonego kolorem żółtym na procesorze 1, w rzeczywistości czas przetwarzania jest proporcjonalnie niewielki, ale można też przyjąć bez straty ogólności, że wynosi on zero).

Zaprezentowany algorytm do optymalizacji problemu szeregowania zadań jest wzorowany na algorytmie ABC ([1]). Pierwszym jego krokiem jest znalezienie rozwiązań początkowych, będących z reguły wynikiem przeszukiwania obszarów rozwiązań wyznaczonych w sposób losowy. Sposób przeszukiwania pojedynczego obszaru wokół wyznaczonej losowo wartości określony jest przez sposób zdefiniowania sąsiedztwa. W problemach kombinatorycznych często określa się dwie permutacje jako sąsiednie, jeżeli różnią się między sobą pozycją dwóch sąsiednich elementów. Stąd przykładowo permutacje 1-**2**-**3**-4 i 1-**3**-**2**-4 są sąsiednie a **1**-2-**3**-4 i **3**-2-**1**-4 już nie. Następnie dla każdej sąsiedniej permutacji wyliczany jest wskaźnik jakości. Jeżeli jego wartość jest większa od obliczonej dla początkowo wyznaczonej, to permutacja ta staje się permutacją bazową (początkową) dla dalszych poszukiwań. Dodatkowo, aby obniżyć koszt wyznaczenia C_{max} sąsiedztwa zastosowano akcelerator [5]. Jeżeli w danym obszarze nie obserwuje się polepszenia wskaźnika, to poszukiwania w nim są przerywane i wyznaczany jest nowy. Kolejną ważną cechą algorytmu jest spamiętywanie testowanych permutacji, w celu uniknięcia ponownego ich wylosowania lub wyznaczenia. Przykładowe wyniki dla 20 maszyn, 500 zadań i 500 iteracji zestawiono w tabeli 1 (implementacja jest rozwinięciem pracy [3]). Czasy przetwarzania prac zostały wygenerowane losowo.

Przykład	Uruchomienie			Optimum	Max. błąd wzgl.
	1	2	3		
1	26429	26469	26511	26189	1.23%
2	26913	26903	26924	26629	1.11%
3	26739	26759	26764	26458	1.16%
4	26780	26750	26679	26549	0.87%
5	26628	26573	26587	26404	0.85%

Tabela. 1 Przykładowe wyniki zaimplementowanego algorytmu.

4. PODSUMOWANIE, DALSZE OBSZARY BADAWCZE

Użycie algorytmu pszczelego do znajdowania optymalnego rozwiązania problemu kombinatorycznego, jakim jest szeregowanie zadań (w szczególności model przepływowy) daje bardzo interesujące i obiecujące wyniki. Jednakże nie należy przy tym zapominać, że czasy poszczególnych prac wchodzących w skład zadań były generowane losowo. Z tego względu dalsze badania powinny skupić się na następujących problemach:

1. Analizy trudnych przypadków danych wejściowych; Należałoby zbudować zbiór takich danych i przeprowadzić testy dla tych danych;
2. Analizy i propozycji wyznaczania sąsiedztwa w inny sposób, jak domyślnie użyty w algorytmie;
3. Analizy i propozycji innych wskaźników jakości (funkcji dopasowania);
4. Analizy warunków początkowych i ich wpływu na znajdywane rozwiązania i ich zbieżność;
5. Analizy wartości początkowych liczności populacji, liczby rozwiązań oznaczonych m i najlepszych spośród nich e, i innych parametrów oraz ich zależności od wielkości i specyfiki zbioru danych wejściowych;
6. Analizy liczby uruchomień i weryfikacji statystycznej średniego błędu popełnianego przez algorytm;

Rozwiązanie tych zagadnień pozwoliłoby na określenie, czy algorytm pszczeli może stać się interesującą alternatywą dla innych algorytmów losowych, w tym genetycznych. Być może również dałoby się określić klasę bądź klasy problemów i powiązanych z nimi zbiorami danych wejściowych, dla których model kolonii pszczół dawałby najlepsze wyniki w ograniczonej liczbie kroków.

Literatura

1. Baykasoglu A., Ozbakor L., Tapkan P., „Artificial Bee Colony and Its Application to Generalized Assignment Problem", I-Tech Education and Publishing, Vienna, pp 29-33, 2007
2. Błazewicz J., Ecker K. H., Schmidt G., Węglarz J., „Scheduling in Computer and Manufacturing Systems", Springer-Verlag, Berlin, 1994
3. Librant S., Skrabacz M., „Zastosowanie algorytmu pszczelego w optymalizacji kombinatorycznej", Państwowa Wyższa Szkoła Zawodowa w Tarnowie, Instytut Politechniczny, praca inżynierska, Tarnów, 2009
4. Pinedo M., "Scheduling – theory, algorithms and systems", Pearson Education, Hong Kong, 2002
5. Smutnicki C., „Algorytmy szeregowania", Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2002
6. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C., „Wprowadzenie do algorytmów", WNT, Warszawa, 2001