MECHANIKA
CZASOPISMO TECHNICZNE
TECHNICAL TRANSACTIONS
MECHANICS
WYDAWNICTWO POLITECHNIKI KRAKOWSKIEJ
4-M/2011
ZESZYT 7
ROK 108
ISSUE 7
YEAR 108

GRZEGORZ FILO*

ZASTOSOWANIE ALGORYTMÓW MRÓWKOWYCH W ROZWIĄZANIU PROBLEMU SZEREGOWANIA ZADAŃ

APPLICATION OF ANT COLONY SYSTEMS IN SOLVING OF TASK SCHEDULING PROBLEM

Streszczenie

W artykule przedstawiono propozycję rozwiązania problemu szeregowania zadań TSP (ang. Task Scheduling Problem) określonego za pomocą skierowanego grafu w warunkach ograniczonych zasobów za pomocą algorytmów mrówkowych ACS (ang. Ant Colony System). Zaprojektowano i wykonano własne oprogramowanie w środowisku RAD C++ Turbo Explorer. Oprogramowanie posłużyło do przeprowadzenia wielu badań pozwalających na dobór parametrów ACS w celu rozwiązania problemu *TSP*. Wyniki wskazują, że zastosowanie ACS stanowi korzystną alternatywę dla używanych algorytmów suboptymalnych typu wielomianowego, jak algorytm Hu lub LPT.

Słowa kluczowe: *algorytmy mrówkowe, optymalizacja, szeregowanie zadań*

Abstract

In this paper is presented proposal of solving the task scheduling problem (TSP) given by directed graph, under limited resources condition using the ant colony system (ACS). The own software in C++ Turbo Explorer rapid application development environment was designed and developed. The software was used to carry out series of simulations, what allowed to choose the best parameters of the ACS in solving of the TSP problem. The results indicate, that using of the ACS is an advantageous option to commonly used suboptimal polynomial algorithms, like Hu or LPT.

Keywords: *ant colony systems, optimization, task scheduling*

* Dr inż. Grzegorz Filo, Instytut Informatyki Stosowanej, Wydział Mechaniczny, Politechnika Krakowska.

Oznaczenia

Z, R – zbiór zadań oraz zbiór zasobów
n, m – liczba zadań oraz liczba zasobów
i, j – indeksy służące do numerowania zadań i zasobów
$\boldsymbol{w}$ – wektor wag (kosztów jednostkowych) wykonania zadań
$\boldsymbol{\tau}$ – macierz czasów wykonania zadań
$\boldsymbol{C}$ – wektor czasów zakończenia wykonywania zadań
ρ, η – natężenie śladu feromonowego, oraz długość (czas wykonania) zadania
α, β – nastawialne parametry
$\boldsymbol{G}_k$, $\boldsymbol{S}_k$, $\boldsymbol{P}_k$ – wektor zadań nieprzydzielonych, aktualnie dostępnych, przydzielonych
γ – współczynnik parowania feromonu
N, q_0 – rozmiar populacji i parametr określający sposób wyboru kolejnego zadania
L – liczba cykli (iteracji)

1. Wstęp

W ostatnich latach, wraz z szybkim zwiększaniem mocy obliczeniowych komputerów, nastąpił wzrost zainteresowania zastosowaniem heurystycznych metod rozwiązywania trudnych problemów kombinatorycznych [7]. Do tej grupy zaliczane są m.in. sieci neuronowe, algorytmy genetyczne, algorytmy mrówkowe, systemy rozmyte czy poszukiwanie rozwiązania metodą Tabu Search [1, 4, 5]. W niniejszym artykule podjęto tematykę rozwiązania problemu szeregowania zadań TSP [2, 6] za pomocą algorytmów mrówkowych. Problem TSP w ogólnym przypadku nie może być rozwiązany optymalnie za pomocą algorytmów deterministycznych, ponieważ pod względem złożoności obliczeniowej należy do grupy NP-zupełnych.

2. Model problemu szeregowania zadań TSP

W ogólności w problemie szeregowania dany jest ustalony system zadań, które powinny być wykonane z wykorzystaniem posiadanego zbioru zasobów. Korzystając z posiadanych charakterystyk zadań i zasobów oraz nałożonych na nie ograniczeń (np. ograniczenie kolejnościowe), należało opracować algorytm optymalizacyjny oparty na systemie kolonii mrówek ACS, którego kryterium jest znalezienie możliwie najkrótszego całkowitego czasu wykonania zadań. Na model problemu składają się modele zasobów, systemu zadań oraz kryterium optymalności [2, 6].

Model zasobów jest zwykle przedstawiany w postaci zbioru $R = \{R_1 \dots R_m\}$. W zależności od problemu poszczególne zasoby mogą być definiowane jako równoległe – spełniające te same funkcje, lub dedykowane – różniące się wykonywanymi funkcjami [2]. Wśród zasobów równoległych można wyróżnić trzy rodzaje, w zależności od wydajności pracy: jeżeli każdy zasób R_i jest w stanie wykonywać wszystkie zadania zdefiniowane w zbiorze Z z taką samą wydajnością, zasoby są określane jako identyczne. Jeżeli poszczególne zasoby różnią się wydajnościami, ale wydajność każdego z nich jest stała, niezależna od zadania, zasoby są określane jako jednorodne. Jeżeli natomiast

wydajności co najmniej dwóch zasobów R_i, R_k zależą od wykonywanego zadania Z_j, wtedy zasoby są niezależne.

Ogólnie, system zadań dla danego zbioru zasobów może być zdefiniowany jako układ $U = (Z, \prec, \tau, R, w)$ [2], gdzie:

- zbiór zadań do wykonania $Z = \{Z_1 \dots Z_n\}$,
- antyzwrotna relacja częściowego porządku $\prec$ zdefiniowana na zbiorze Z określa operacyjne ograniczenia kolejnościowe ($Z_i \prec Z_k$ oznacza, że wykonanie zadania Z_k nie może się rozpocząć przed zakończeniem wykonywania zadania Z_i). Istnienie przynajmniej jednego ograniczenia kolejnościowego sprawia, że zbiór Z staje się zbiorem zadań zależnych,
- macierz czasów wykonania zadań $\boldsymbol{\tau}$ o wymiarze $m \times n$, gdzie $\boldsymbol{\tau}_{ij} > 0$ jest czasem wykonywania zadania Z_j, $1 \le j \le n$, z wykorzystaniem zasobu R_i, $1 \le i \le m$. Zakłada się, że $\boldsymbol{\tau}_{ij} = \infty$, jeżeli zadanie Z_j nie może być wykonywane przez zasób R_i oraz że dla każdego j istnieje co najmniej jedno i dla którego $\boldsymbol{\tau}_{ij} < \infty$. Jeżeli zasoby są identyczne, $\boldsymbol{\tau}_j$ oznacza czas wykonywania zadania Z_j na dowolnym zasobie,
- wagi w_i, $1 \le i \le n$ są interpretowane jako koszty jednostkowe, które w ogólności mogą być dowolnymi funkcjami własności uszeregowania mającymi wpływ na zadanie Z_i. Zwykle przyjmuje się, że w_i są stałe, dlatego koszt związany z zakończeniem wykonywania zadania Z_i w chwili t jest równy $w_i \cdot t$.

Jako kryterium optymalności przyjęto minimalizację długości czasu wykonania wszystkich zadań $MIN(C_{max})$, określonego za pomocą zadania o największym czasie zakończenia wykonywania $\max\{C_j\}$:

$$MIN(C_{max}) = MIN(\max\{C_j\}, \quad j = 1 \dots n). \tag{1}$$

3. Implementacja algorytmów obliczeniowych

W zbudowanym oprogramowaniu zaimplementowano algorytm ACS oraz algorytmy LPT i Hu, które posłużyły do przeprowadzenia badań porównawczych.

3.1. Model systemu kolonii mrówek ACS

Algorytmy mrówkowe to grupa metod przeznaczona do poszukiwania optymalnego rozwiązania zadań. Metody te, zaproponowane po raz pierwszy w latach 90., stosowane są zazwyczaj do rozwiązywania trudnych problemów kombinatorycznych [3, 4]. Podstawowymi cechami algorytmów ACS są: poszukiwanie oparte na populacji osobników, które nie komunikują się w sposób bezpośredni, odporność na nieznane warunki środowiska i wzajemne stymulacja przez dodatnie sprzężenie zwrotne.

ACS jest systemem wieloagentowym, w którym działanie poszczególnych agentów odpowiada działaniu pojedynczych mrówek poszukujących wartościowych dla nich zasobów. Każdy z agentów, zwanych, sztucznymi mrówkami, realizuje identyczną strategię szukania najkrótszej drogi do celu. Ponadto istnieje w systemie pewna forma gromadzenia doświadczeń i uwzględnienia ich w dalszych poszukiwaniach (ślad feromonowy), co sprawia, że w miarę upływu czasu mrówki wspólnie dopracowują się zbioru najkrótszych

ścieżek wiodących ich do wyznaczonych celów. Schemat działania algorytmu mrówkowego opracowanego do rozwiązania problemu TSP zapisano następująco:

```
void SystemMrowkowy()
{
  N = ustal_rozmiar_populacji();
  ustal_poczatkowy_poziom_feromonu();
  do{
    for i=0 to N-1 wygeneruj_sciezke_mrowki(i);
    wyznacz_najlepszy_wynik();
    odparuj_i_uaktualnij_feromon();
  }while(warunek_konca_algorytmu);
}
```

Na początku algorytmu tworzona jest populacja mrówek o zadanym rozmiarze N i ustalany początkowy poziom feromonu. W każdej iteracji (cyklu) dany osobnik rozpoczyna wędrówkę od zadania początkowego. Wszystkie zadania oprócz pierwszego są zapisywane w wektorze $\boldsymbol{G}_\mathbf{k}$ – zadań nieprzydzielonych. Następnie następuje wyznaczenie wektora zadań dostępnych w bieżącym kroku $\boldsymbol{S}_\mathbf{k}$ i osobnik wykonuje krok poprzez wybór jednego z nich. Algorytm wyboru jest zależny od parametru q_0. W zależności od wartości tego parametru może nastąpić losowy wybór zadania z wektora $\boldsymbol{S}_\mathbf{k}$ lub metodą ruletki, przy czym w tym wypadku prawdopodobieństwo wyboru wierzchołka jest dane wzorem:

$$p_{ij}^{k}(t) = \frac{[\rho_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_j^k} ([\rho_{il}(t)]^{\alpha}[\eta_{il}]^{\beta}}, \quad \text{gdzie} \quad j \in S_k^i. \tag{2}$$

Po wyborze zadania jest ono przyporządkowywane do zasobu, na którym może być wykonane najwcześniej przy uwzględnieniu ograniczeń. Przyporządkowane zadanie jest usuwane z wektora $\boldsymbol{G}_\mathbf{k}$ zapisywane w pamięci osobnika (wektor $\boldsymbol{P}_\mathbf{k}$) w celu uniknięcia jego ponownego wyboru i krok jest powtarzany do momentu aż $\boldsymbol{G}_\mathbf{k} = \varnothing$. Po wygenerowaniu ścieżek przez wszystkie osobniki populacji wyznaczana jest najkrótsza z nich (najlepsze rozwiązanie). Następuje odparowanie feromonu ze wszystkich krawędzi, a następnie dodanie nowej wartości na krawędziach, po których poruszały się osobniki w bieżącej iteracji według wzoru:

$$\rho_{ij}(t+1) = (1-\gamma)\cdot\rho_{ij}(t) + \sum_{k=1}^{m}\Delta\rho_{ij}(t) \tag{3}$$

Cykle są powtarzane, aż zostanie spełniony warunek końca algorytmu. Warunkiem tym może być osiągnięcie zadanej liczby cykli lub uzyskanie czasu uszeregowania krótszego niż założony czas wymagany.

3.2. Algorytm LPT I Hu

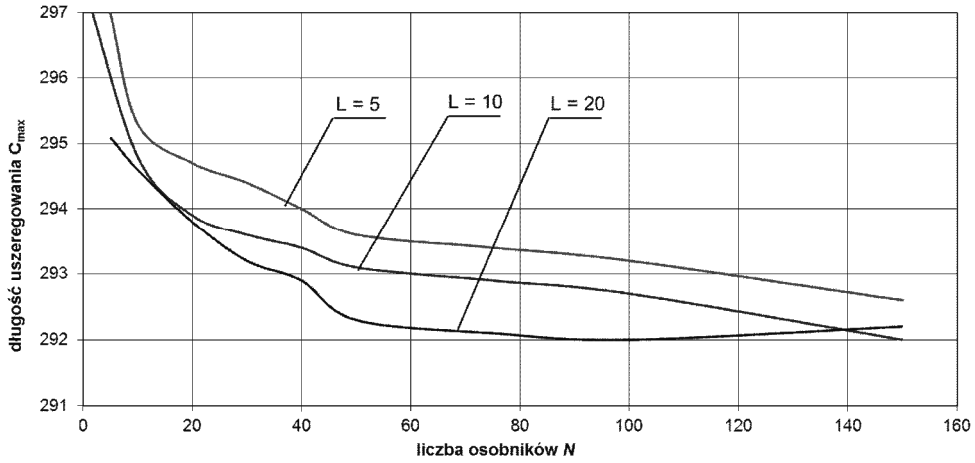
Algorytm LPT jest typu wielomianowego, suboptymalnego. Polega on na przydzielaniu aktualnie wolnego zasobu do wykonania najdłuższego zadania z listy dostępnych w danej chwili. W ocenie jego dokładności udowodniono [2], że dla zbioru zadań niezależnych stosunek długości uszeregowania otrzymanego za pomocą algorytmu LPT do długości odpowiedniego uszeregowania optymalnego spełnia nierówność:

$$\frac{C_{\max}^{\mathrm{LPT}}}{C_{\max}^{*}} \le \frac{4}{3} - \frac{1}{3m}. \tag{4}$$

Algorytm Hu jest algorytmem dokładnym, wielomianowym dla problemu szeregowania zadań o jednostkowych czasach wykonania tworzących diagraf typu „antydrzewa" przy założeniu identycznych zasobów. Algorytm ten składa się z trzech kroków [2]. Na początku następuje określenie poziomów zadań. Jeśli liczba zadań bez poprzedników jest mniejsza lub równa m, następuje przydzielenie zadaniom zasobów i przejście do następnego kroku. W przeciwnym razie następuje wybór m zadań o najwyższych poziomach i przydzielenie im zasobów (dla zadań tego samego poziomu wybór jest dowolny). Na końcu kroku przydzielone zadania są usuwane z grafu. Krok drugi należy powtarzać do momentu przydzielenia wszystkich zadań. Poziom zadania w grafie określa długość najdłuższej drogi rozpoczynającej się w danym wierzchołku, liczonej jako suma czasów wykonania zadań leżących na tej drodze.

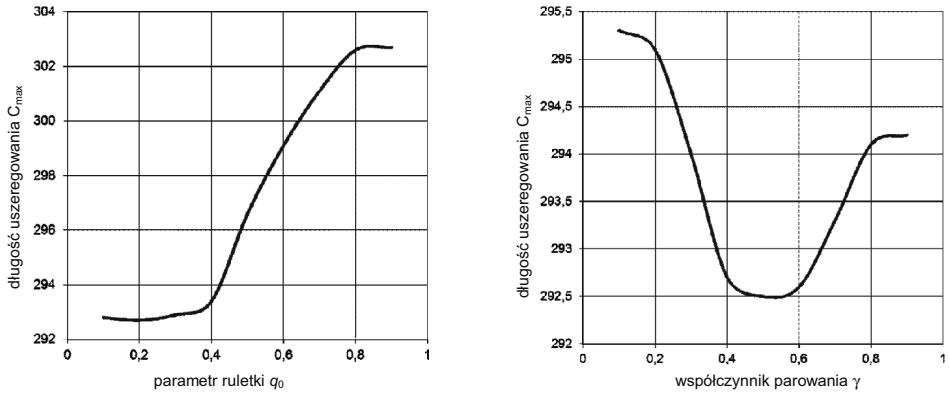
4. Charakterystyka i wyniki badań

W badaniach wykorzystano testowe grafy zadań dostępne w serwisie internetowym [8]. Dostępne są grafy o rozmiarze od n = 50 do n = 5000. Na rysunku 1 przedstawiono zależność uzyskanej długości uszeregowania zadań przykładowego grafu (n = 100) w zależności od rozmiaru populacji ACS i liczby cykli obliczeń. Na wykresie zamieszczono średnie wartości uzyskane z 10 powtórzeń każdej symulacji. Parametry ACS: α = 1,0, β = 1,0, γ = 0,5, q_0 = 0,2. Długość uszeregowania algorytmem LPT wyniosła $C_{\max}^{\mathrm{LPT}} = 317$.



Rys. 1. Długość uszeregowania w funkcji liczby osobników N i liczby cykli L
Fig. 1. Length of scheduling time in function of agents number N and iterations number L

Na rysunku 2 pokazano zależność uzyskiwanej długości uszeregowania od wartości parametrów q_0 i γ dla tego samego grafu, przy α = 1,0, β = 1,0, N = 100.



Rys. 2. Zależność długości uszeregowania w funkcji parametrów q_0 i γ
Fig. 2. Length of scheduling time in function of parameters q_0 and γ

5. Wnioski

W artykule podjęto zadanie rozwiązania problemu szeregowania zadań TSP za pomocą algorytmów mrówkowych ACS. Zadanie wykonano za pomocą własnego oprogramowania wykonanego w języku C++. Wyniki badań wskazują, że zastosowanie ACS daje znacząco lepsze wyniki niż suboptymalny algorytm LPT. Na podstawie uzyskanych wyników zaproponowano dobór wybranych parametrów ACS na potrzeby rozwiązywania problemu TSP: parametr ruletki $q_0 = 0{,}2$ oraz współczynnik parowania feromonu $\gamma = 0{,}5$.

Literatura

[1] Anguliar J., *A General Ant Colony Model to solve Combinatorial Optimization Problems*, Revista Colombiana de Computacion, Vol. 2, No. 1, 2001.
[2] Coffman E., *Teoria szeregowania zadań*, WNT, Warszawa, 1980.
[3] Corne D., Dorigo M., Glove F., *New ideas in Optimization*, McGraw Hill, 1999.
[4] Dorigo M., Stützle T., *Ant colony optimization*, MIT Press, USA, 2004.
[5] Grzymkowski R. i inni, *Wybrane algorytmy optymalizacji. Algorytmy genetyczne*, *Algorytmy mrówkowe*, PKJS, 2008.
[6] Sinnen O., Sousa L., *On Task Scheduling Accuracy: Evaluation Methodology and Results*, The Journal of Supercomputing, Vol. 27, No. 2, Kluwer 2004.
[7] Trojanowski K., *Metaheurystyki praktycznie*, Wydawnictwo Wyższej Szkoły Informatyki i Zarządzania, Warszawa 2003.
[8] Serwis zawierający przykładowe grafy zadań (kasahara.elec.waseda.ac.jp/schedule/index.html).