Projektowanie Efektywnych Algorytmów

Projekt

W ramach projektu należy zrealizować trzy poniższe zadania. Każde zadanie będzie oceniane oddzielnie. Ocena końcowa z projektu będzie średnią arytmetyczną z ocen cząstkowych. Zadanie 4. jest zadaniem na ocenę celującą. Jednak, uzyskanie oceny celującej wymaga uzyskania z pozostałych zadań co najmniej 14,0 punktów (np. 4,5 4,5 5,0).
Każde zadanie polega na zaimplementowaniu i przetestowaniu algorytmu mateheurystycznego dla jednego z dwóch poniższych problemów. Przy czym wszystkie zadania muszą dotyczyć tego samego problemu. W kolejnych sprawozdaniach należy porównać algorytmy między sobą, tzn. porównać co najmniej ich czasy działania oraz uzyskiwane wartości funkcji celu.
Projekty realizowane są indywidualnie.

Problem 1: Problem komiwojażera
Powinien być wszystkim znany.
Na stronie
comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
umieszczono przykłady testowe dla tego problemu, których należy użyć przy testowaniu algorytmów (dla symetrycznego i niesymetrycznego problemu).

Problem 2: Jednoprocesorowy problem szeregowania zadań przy kryterium minimalizacji ważonej sumy opóźnień zadań
Problem ten może być opisany następująco:
Danych jest n zadań (o numerach od 1 do n), które mają być wykonane bez przerwań przez pojedynczy procesor mogący wykonywać co najwyżej jedno zadanie jednocześnie. Każde zadanie j jest dostępne do wykonania w chwili zero, do wykonania wymaga $p_j > 0$ jednostek czasu oraz ma określoną wagę (priorytet) $w_j > 0$ i oczekiwany termin zakończenia wykonania $d_j > 0$. Zadanie j jest spóźnione, jeżeli zakończy się wykonywać po swoim terminie d_j, a miarą tego opóźnienia jest wielkość $T_j = max(0, C_j, -d_j)$, gdzie C_j jest terminem zakończenia wykonywania zadania j. Zadanie polega na znalezieniu takiej kolejności

wykonywania zadań (permutacji), aby zminimalizować kryterium $TWT = \sum_{j=1}^{n} w_j * T_j$. Na stronie

http://people.brunel.ac.uk/ mastjjb/jeb/orlib/wtinfo.html

umieszczono przykłady testowe dla tego problemu, których należy użyć przy testowaniu algorytmów.

Zadanie 1. Metoda **programowania dynamicznego lub podziału i ograniczeń** (*ang. dynamic programming, branch and bound*).
Należy zaimplementować wybraną metodę dla jednego z podanych powyżej problemów oraz wykonać testy polegające na pomiarze czasu działania algorytmu w zależności od wielkości instancji oraz jakości dostarczanych rozwiązań. Należy porównać rozwiązanie dostarczone przez algorytm z najlepszymi znanymi rozwiązaniami dla przykładów testowych.

Termin oddania zadania to 8 listopada 2017 roku. Chyba, że prowadzący ustali inny.

Zadanie 2. Algorytm przeszukiwania z zakazami (*ang. tabu search*)
Należy zrobić to samo co w zadaniu 1 (dla tego samego problemu!). Dodatkowo, należy porównać wyniki z wynikami z zadania 1. Wskazane jest, aby algorytm uwzględniał mechanizm dywersyfikacji przeszukiwania przestrzeni rozwiązań (powroty, ruchy losowe, itp.).

Termin oddania zadania to 13 grudnia 2017 roku. Chyba, że prowadzący ustali inny.

Zadanie 3. Algorytm genetyczny (*ang. genetic algorithm*)
Należy zrobić to samo co w zadaniach 1 i 2 (dla tego samego problemu!). Dodatkowo, należy porównać wyniki z wynikami otrzymanymi w poprzednich zadaniach. Wskazane jest, aby algorytm uwzględniał zaawansowane mechanizmy (koewolucję, ewolucję wyspową, samodoskonalenie osobników, itp.).

Termin oddania zadania to 17 stycznia 2018 roku. Chyba, że prowadzący ustali inny.

Zadanie 4. Do wyboru: **algorytm mrówkowy** (*ang. ant colony algorithm*), **algorytm przeszukiwania rozproszonego** (*ang. scatter search*), **sztuczna sieć neuronowa** (*ang. artificial neural network*), **algorytm bazujący na sztucznym systemie immunologicznym** (*ang. artificial immune system*). Szczegóły dotyczące tego zadania należy konsultować z prowadzącym.

Literatura do zadań 1-3

1. D.E. Goldberg, Algorytmy genetyczne i ich zastosowania, Warszawa, WNT 1998.

2. A. Janiak, Wybrane problemy i algorytmy szeregowania zadań i rozdziału zasobów, PLJ 1999.

3. Z Michalewicz, Algorytmy genetyczne + struktury danych = programy ewolucyjne, Warszawa, WNT 1996.

4. Z Michalewicz, D.B. Fogel, Jak to rozwiązać, czyli nowoczesna heurystyka, WNT 2006.

5. C. Smutnicki, Algorytmy szeregowania, EXIT 2002.

6. V. Cerny, A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm, Journal of Optimization Theory and Applications, 45: 41-51, 1985.

7. H.A.J. Crauwels, C.N. Potts, L.N. Van Wassenhove, Local search heuristics for the single machine total weighted tardiness scheduling problem. Informs Journal on Computing, 10: 342-350, 1988.

Literatura do zadania 4.

1. M. Dorigo, T. Stuuzle, Ant Colony Optimization, MIT Press 2004.

2. M. Dorigo, G. Di Caro, L.M. Gambardella, Ant Algorithms for Discrete Optimization. Artificial Life, 5(2): 137-172, 1999.

3. S.T. Wierzchoń, Sztuczne systemy immunologiczne. Teoria i Zastosowania, EXIT 2001.

4. L.N. De Castro, J.I. Timmis, Artificial Immune Systems as a Novel Soft Computing Paradigm, Soft Computing Journal, vol 7, 2003.

5. M. Laguna, R. Marti, R.C. Marti, Scatter search: methodology and implementation in C, Kluwer 2003.