Projektowanie efektywnych algorytmów – projekt – zasady zaliczania

W ramach zajęć należy zrealizować przedstawione przez prowadzącego zadania projektowe. Zadania realizowane są w grupach jednoosobowych.

Z wykonania każdego zadania należy sporządzić sprawozdanie. Zawartość sprawozdania jest opisana w ramach każdego zadania. Na pierwszej stronie sprawozdania MUSZĄ być podane następujące informacje: imię, nazwisko i numer indeksu autora, nr zadania (zadania są numerowane od 1 do 3) oraz przynależność do grupy projektowej (termin np. piątek 9.00-11.00) wraz z nazwiskiem prowadzącego.

Sprawozdanie należy oddać PROWADZĄCEMU w wersji papierowej. Do sprawozdania należy dołączyć wersję elektroniczną programów (kody + wersja wykonywalna).

Proponuje się następujące terminy oddawania sprawozdań z poszczególnych zadań projektowych (w szczególnych przypadkach prowadzący może zmienić podane terminy):

1. 16 listopad 2017
2. 18 grudzień 2017
3. 15 styczeń 2018

Każdy tydzień spóźnienia oznacza obniżenie oceny o 0.5! Oczywiście zadanie można oddać wcześniej (za oddanie wcześniej zwiększa się ocenę o 0,25 a za jeszcze wcześniej o 0,5), ale aby dostać ocenę na koniec 5,5 należy uzyskać sumaryczną ocenę z wszystkich projektów równą 14pkt (oceny bez bonusów za termin) oraz zrobić dodatkowe zadanie. Ocena za pojedyncze zadanie projektowe wystawiana jest w sposób ciągły od 2 do wartości maksymalnej (określonej przez wariant zadania)

Ocena końcowa z projektu stanowi średnią arytmetyczną ocen z poszczególnych zadań z uwzględnieniem spóźnień wg wzoru: (z1+z2+z3) / 3

Warunkiem koniecznym zaliczenia projektu jest uzyskanie oceny pozytywnej oceny końcowej (tzn. >=3.0). Ostateczny termin oddania zadania pierwszego upływa na tydzień przed terminem oddania zadania drugiego. Ostateczny terminem oddania zadania drugiego upływa tydzień wcześniej przed oddaniem zdania trzeciego. Ostateczny termin zadania trzeciego wyznacza data ostatnich zajęć. Nieoddanie zadania w w/w terminach skutkuje niezaliczeniem przedmiotu.

Zajęcia z projektu trwają 2 godziny lekcyjne bez przerwy - 1,5 godziny zegarowej.

Wszelkie sprawy z projektem załatwiamy tylko na godzinach projektu (a nie konsultacji).

Przy oddawaniu projektu wymagana jest znajomość problemu (student będzie odpytywany z tej znajomości), którego dotyczy jak i programu. **Jeżeli student zostanie złapany na nieznajomości programu, który oddaje, to wówczas traktuje się to jako fałszerstwo i stawia się ocenę niedostateczną na zaliczenie (prowadzący nie ma prawa wiedzieć więcej niż student na temat jego programu)**. Podobnie traktuje się osobę, która odda program prawie identyczny jak ktoś wcześniej. Twierdzenie, iż ktoś pomagał przy pisaniu nie jest wytłumaczeniem. Zabrania się kopiować programów z internetu.

Ogólnie o zadaniach z PEA

W ramach projektu należy zrealizować trzy poniższe zadania. Każde zadanie polega na zaimplementowaniu i przetestowaniu algorytmu dla jednego z dwóch poniższych problemów. Przy czym wszystkie zadania muszą dotyczyć tego samego problemu. W kolejnych sprawozdaniach należy porównać algorytmy miedzy sobą, tzn. porównać ich czasy działania oraz uzyskiwane wartości funkcji celu.

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

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 maja 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) $wj > 0$ i oczekiwany termin zakończenia wykonania $dj > 0$. Zadanie j jest spóźnione, jeżeli zakończy się wykonywać po swoim terminie dj , a miarą tego opóźnienia jest wielkość $Tj = max(0,Cj - dj)$, gdzie Cj jest terminem zakończenia wykonywania zadania j. Zadanie polega na znalezieniu takiej kolejności wykonywania zadań (permutacji), aby zminimalizować kryterium $TWT =$

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

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 algorytmem symulowanego wyżarzania. Wskazane jest, aby algorytm uwzględniał mechanizm dywersyfikacji przeszukiwania przestrzeni rozwiązań (powroty, ruchy losowe, itp.). Należy porównać rozwiązanie dostarczone przez algorytm z najlepszymi znanymi rozwiązaniami dla przykładów testowych.

Zadanie 3. Algorytm genetyczny (ang. genetic algorithm)

Należy zrobić to samo co w zadaniu 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.).

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 zadan 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. Wierzchon, 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.