ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994
Seria: AUTOMATYKA z.114 Nr kol. 1250

Konrad WALA, Henryka GĄDEK-MADEJA
Akademia Górniczo-Hutnicza

ALGORYTM PRZYBLIŻONY DLA WIELOPOZIOMOWEGO UOGÓLNIONEGO ZAGADNIENIA PRZYDZIAŁU

Streszczenie: W pracy przedstawiono model matematyczny wielopoziomowego zagadnienia przydziału. W ogólności. problem ten polega na przydziale n zadań do m rodzajów zasobów oraz maksymalnie l poziomów efektywności. Zaprezentowano algorytm przybliżony opracowany na bazie algorytmu genetycznego. Wyniki badań eksperymentalnych wskazują, że zaproponowany algorytm wyznacza zupełnie dobre rozwiązania dla zagadnień o rozmiarze spotykanym w praktyce.

APPROXIMATE ALGORITHM FOR MULTILEVEL GENERALIZED ASSIGNMENT PROBLEM

Summary: This paper considers a mathematical model of the multilevel generalized assignment problem. In general, this problem consists in assigning n tasks to m agents with a maximum of l efficiency levels. An aproximate algorithm developed on the base of genetic algorithm is presented. Empirical experimentations indicate that the proposed algorithm produces quite good solutions for practical-size problems.

АППРОКСИМАЦИОННЫЙ АЛГОРИТМ ДЛЯ МНОГОУРОВНЕВОЙ ОБОБЩЕННОЙ ПРОБЛЕМЫ НАЗНАЧЕНИЯ

Резюме: В роботе ставлена модел многоуровневой и обобщенной проблемы назначения. В общих чертах, проблема назначения заключается в назначении n задач к m сортам ресурсов и максимльно l уровней эффективности. Предлагается аппроксимационный алгоритм разработанный на основе генетического алгоритма. Результаты экспериментальных испитании показывают что предложенный алгоритм определяет доброкачественные решния для проблем практического размера.

1. Wprowadzenie

Rozważając zagadnienie modelowania przydziału zadań w dużych firmach produkcyjnych, po raz pierwszy w roku 1979 Glover, Hultz i Klingman [2] opisali wielopoziomowe zagadnienie przydziału MGAP (ang. multilevel generalized assignment problem). Zagadnienie MGAP różni się od standardowego uogólnionego zagadnienia przydziału GAP (ang. generalized assignment problem; por. praca przeglądowa [7]) możliwością wykonania zadań na wielu poziomach wydajności. Występowanie w modelu dużej liczby 0-1 zmiennych decyzyjnych uniemożliwia, dla zagadnień występujących w praktyce, znalezienie rozwiązania optymalnego za pomocą komercyjnych pakietów optymalizacji. W pracy [6] zaproponowano skuteczny algorytm przybliżony rozwiązania zagadnienia MGAP bazujący na metodzie przeszukiwania otoczenia, zwanej "tabu search". Pracując nad efektywnym algorytmem przybliżonym rozwiązywania zagadnienia MGP na komputerach klasy PC próbowalismy, na początku bez powodzenia, oprzeć się właśnie na tym algorytmie. Dopiero eksperymenty związane z różnymi modyfikacjami algorytmu genetycznego dały zachęcające wyniki.

2. Model matematyczny zagadnienia MGAP

W ogólnym przypadku, model MGAP opisuje zagadnienie przydziału n zadań do m rodzajów zasobów (środków, ang. agents) przy istnieniu, w maksymalnym przypadku, l poziomów wydajności. Zakłada się, że każde zadanie j jest przydzielone dokładnie do jednego rodzaju zasobu i (krótko: zasobu i) oraz jednego poziomu wydajności k. Do każdego zasobu i może być przydzielone więcej niż jedno zadanie, ale suma zapotrzebowania tych zadań na zasób nie może przekroczyć wielkości b_i dostępnych ilości zasobu i. Wielkość zasobu potrzebna w przypadku przydziału zadania j do zasobu i przy poziomie k oznaczona jest przez a_{ijk}, $a_{ijk} > 0$, natomiast odpowiedni koszt przydziału jest równy c_{ijk}, $c_{ijk} > 0$. Zwykle w zagadnieniach praktycznych, dla każdego przydziału j-i, zachodzi następujący związek pomiędzy kosztem i potrzebnym zasobem: jeżeli $a_{ijk'} < a_{ijk''}$ to $c_{ijk'} > c_{ijk''}$.
Ponadto zakłada się, że $a_{ijk} = d$ w przypadku, gdy przydział zadania j do zasobu i oraz poziomu k jest niedopuszczalny, gdzie d, $d > \max_i b_i$, jest dostatecznie dużą liczbą. Po wprowadzeniu binarnych zmiennych decyzyjnych $x=[x_{ijk}]_{m*n*l}$, gdzie x_{ijk} jest równe 1 w przypadku przydziału zadania j do zasobu i przy poziomie k, powyższe kombinatoryczne zagadnienie optymalizacji można sformalizować w następujący sposób.

Należy zminimalizować globalny koszt przydziału:

$$z(x) = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{l} c_{ijk}x_{ijk} \qquad (1)$$

przy następujących ograniczeniach:

$$\sum_{i=1}^{m}\sum_{k=1}^{l} x_{ijk} = 1, \qquad j=1,\dots,n \qquad (2)$$

$$\sum_{j=1}^{n}\sum_{k=1}^{l} a_{ijk}x_{ijk} \le b_i, \qquad i=1,\dots,m \qquad (3)$$

$$x_{ijk} \in \{0,1\}, \qquad i=1,\dots,m;\ j=1,\dots,n;\ k=1,\dots,l; \qquad (4)$$

W powyższym modelu ograniczenia (2) zapewniają przydział każdego zadania do dokładnie jednego wykonawcy i poziomu, natomiast ograniczenia (3) uwzględniają dostępne wielkości zasobów.

W zastosowaniu opisanym w [2] celem postawionego zagadnienia jest minimalizacja łącznych kosztów produkcji i magazynowania na drodze określenia optymalnego rozmiaru partii wyrobów i optymalnego przydziału partii wyrobów do maszyn. W systemie wytwarzania istnieje maksymalnie l możliwych rozmiarów partii wyrobów oraz maszyny pracujące równolegle o różnej wydajności i koszcie operacyjnym. Niektóre z maszyn są maszynami ogólnego zastosowania zdolnymi do wytwarzania wielu gatunków wyrobów, podczas gdy inne mogą być bardziej wyspecjalizowane. W tym przykładzie c_{ijk} reprezentuje łączne koszty uruchomienia produkcji, produkcji i magazynowania (na jednostkę czasu), w przypadku gdy wyrób j, realizowany w partii o rozmiarze k, jest przydzielony do maszyny i. Zwykle dla pary wyrób-maszyna realizacja partii małego rozmiaru jest związana z dużymi kosztami łącznymi i odwrotnie. Po wprowadzeniu funkcji kary za przekroczenie ograniczeń zasobowych (3) o postaci:

$$V(x) = \sum_{i=1}^{m} \left| \min\{0, s_i\} \right|, \qquad (5)$$

gdzie: $s_i = b_i - \sum_{j=1}^{n}\sum_{k=1}^{l} a_{ijk}x_{ijk}$

przyjmujemy, do dalszych rozważań, następującą funkcję oceny rozwiązań:

$$f(x) = z(x) + \alpha V(x), \qquad (6)$$

gdzie α ($\alpha \ge 0$) jest współczynnikiem funkcji kary.

Algorytmy genetyczne wymagają reprezentacji rozwiązania w postaci ciągu. Przyjęto, że rozwiązanie zagadnienia MGAP jest sformalizowane w postaci następującego ciągu dwójek:

$\Pi = (\Pi(1), \Pi(2),\dots,\Pi(j),\dots,\Pi(n))$, gdzie $\Pi(j)=(i,k)$ określa przydział zadania j do wykonawcy i na poziomie k, tj. $\Pi(j) = (i,k)$ jest równoważne $x_{ijk} = 1$.

3. Algorytm genetyczny

Zaproponowany w 1975 roku przez Hollanda [4] algorytm maszynowego uczenia się rozwiązywania zagadnień, których rozwiązanie jest sformalizowane w postaci ciągu binarnego, jest nazywany algorytmem genetycznym GA (ang. genetic algorithm). W następnych latach algorytm GA zaimplementowano do rozwiązania imponującej liczby różnorodnych zagadnień (por. monografie [3], [8]). Powszechnie, oprócz stosowania ciągów binarnych, zaczęto także formalizować rozwiązania za pomocą skończonych ciągów określonych na wieloelementowym alfabecie.

Ogólnie, algorytm genetyczny jest iteracyjną procedurą poprawiania rozwiązań zawierającą zbiór POPULACJA najlepszych znanych rozwiązań zagadnienia oraz zbiór pseudogenetycznych operatorów nazywanych krótko operatorami genetycznymi. Na początku pracy algorytmu rozwiązania zbioru POPULACJA są wyznaczane na drodze losowania lub za pomocą specjalnej procedury heurystycznej. Przyjęto, że zbiór POPULACJA ma stały rozmiar, tj. liczba M rozwiązań znajdujących się w zbiorze POPULACJA nie zmienia się podczas procesu optymalizacji. Eksploatowany przez nas, już we wcześniejszych pracach [1], [5], [9], proces optymalizacji pod nazwą algorytmu genetycznego GA, można ogólnie opisać za pomocą następujących kroków.

Algorytm GA

Krok 1. Wyznaczenie populacji początkowej.

Wygeneruj, w sposob losowy, μM rozwiązań. W zbiorze POPULACJA umieść M najlepszych, biorąc pod uwagę funkcję oceny $f(\Pi)$, rozwiązań.

Krok 2. Wybór operatora genetycznego.

Wylosuj numer operatora genetycznego (każdy z sześciu operatorów losowany jest z prawdopodobieństwem p_r, $r = 1...6$; przy czym $p_1 + p_2 + ... + p_6 = 1$ oraz $p_1 \geq 0$, $p_2 \geq 0, ..., p_6 \geq 0$).

Krok 3. Wybór rodzica/rodziców.

Dla wybranego operatora wylosuj zgodnie z rozkładem równomiernym, ze zbioru POPULACJA, zależnie od typu operatora, jedno lub dwa rozwiązania, które nazywamy rodzicami .

Krok 4. Generowanie potomków.

Za pomocą operatora genetycznego dokonaj modyfikacji rozwiązania-rodzica /rozwiązań-rodziców i wyznacz w ten sposób rozwiązanie-potomek / rozwiązania-potomki.

Krok 5. Poprawa rozwiązań zbioru POPULACJA.

Dla każdego rozwiązania-potomka wykonaj: oblicz wartość funkcji oceny $f(\Pi_{pot})$ potomka Π_{pot} i jeżeli wartość ta jest mniejsza od funkcji oceny najgorszego rozwiązania

Π_{max} populacji (tj. $f(\Pi_{pot})<f(\Pi_{max})$, gdzie $f(\Pi_{max})=\max\{f(\Pi): \Pi \in POPULACJA\}$), to umieść potomka Π_{pot} w zbiorze POPULACJA usuwając zarazem z tego zbioru najgorsze rozwiązanie Π_{max}.

Krok 6. STOP.

Jeżeli wygenerowano zadaną liczbę L potomków, to STOP, wydrukuj najlepsze rozwiązanie zbioru POPULACJA. W przeciwnym przypadku idź do kroku 2.

Wielkości μ, M, L, $p_1,...,p_6$ są parametrami algorytmu. Współczynnik μ ($\mu \geq 1$) nazywamy współczynnikiem wstępnej selekcji rozwiązań. Jeżeli $\mu>1$, to, w kroku 1, po wyznaczeniu μM rozwiązań następuje wstępna selekcja rozwiązań: M najlepszych jest umieszczanych w zbiorze POPULACJA, natomiast $(\mu-1)M$ gorszych zostanie odrzuconych.

Oznaczmy dalej przez D(j) zbiór możliwych przydziałów zadania j do wykonawcy i przy poziomie wydajności k, $D(j)=\{(i,k): a_{ijk}< d\}$. Wyznaczenie rozwiązań początkowych w kroku 1 algorytmu polega na zastosowaniu procedury RANDOM_UNIFORM losowania, o równomiernym rozkładzie, która dla każdego zadania j wyznacza możliwy przydział: $\Pi(j)=(i,k)=$ RANDOM_UNIFORM(D(j)).

Oczywiście, zwykle tak wyznaczone rozwiązania początkowe naruszają ograniczenia (3), tj. zwykle funkcja kary V takich rozwiązań jest dodatnia.

Zauważyliśmy podczas badań eksperymentalnych, że korzystne, dla efektywności procesu szukania, jest zwiększenie liczby operatorów genetycznych. Wydaje się, że różnorodność procedur poprawiania rozwiązań, jakimi są operatory genetyczne, zapewnia wyznaczanie rozwiązań z różnych obszarów i tym samym przeszukiwanie tych obszarów. Standardowe operatory genetyczne uzupełniliśmy prostymi procedurami ograniczonego przeglądu wariantów rozwiązań (por.[5,9]) oraz zastosowaliśmy, jako operator genetyczny, najprostszy algorytm optymalizacji lokalnej. Proponowany algorytm jest wyposażony w następujące operatory genetyczne.

1. Operator optymalizacji lokalnej.

Operator 1 wyznacza jednego potomka Π^1 na podstawie jednego rodzica Π za pomocą instrukcji:

(a) Określ otoczenie $S(\Pi) = \{ \Pi' : \Pi' = (\Pi(1), \Pi(2),...,\Pi'(j),...,\Pi(n)), \Pi'(j) \in D(j), \Pi'(j) \neq \Pi(j)\}$

(b) Spróbuj znaleźć rozwiązanie $\Pi' \in S(\Pi)$, takie, że $f(\Pi')<f(\Pi)$.

(c) Jeżeli znaleziono Π' (pierwsze poprawiające), to podstaw $\Pi:=\Pi'$ i przejdź do instrukcji (a), w przeciwnym przypadku przejdź do instrukcji (d).

(d) Podstaw $\Pi^1:=\Pi$.

2. Operator inwersji z przeszukiwaniem.

Operator 2 wyznacza jednego potomka Π^1 na podstawie jednego rodzica Π za pomocą instrukcji:

(a) Wybierz liczbę naturalną i, $1 \le i \le n$, oraz najlepszą liczbę j,
$j = \arg\min_{j \ne i} \{ f(\Pi(1),\Pi(2),...,\Pi(i-1),\Pi(j),\Pi(i+1),...,\Pi(j-1),\Pi(i),\Pi(j+1),...,\Pi(n)\}$.

(b) Przestaw element $\Pi(i)$ rodzica z elementem $\Pi(j)$ wyznaczając w ten sposób potomka:
$\Pi^1 = (\Pi(1),\Pi(2),...,\Pi(i-1),\Pi(j),...,\Pi(j-1),\Pi(i),...,\Pi(n))$.

(3) Jednopunktowy operator krzyżowania z przeszukiwaniem.

Operator 3 wyznacza dwa potomki:
$\Pi^1=(\Pi'(1),\Pi'(2),...,\Pi'(j),\Pi''(j+1),...\Pi''(n))$,
$\Pi^2=(\Pi''(1),\Pi''(2),...,\Pi''(j),\Pi'(j+1),...\Pi'(n))$
na podstawie dwóch rodziców:
$\Pi'=(\Pi'(1),\Pi'(2),...,\Pi'(j),\Pi'(j+1),...,\Pi'(n))$,
$\Pi''=(\Pi''(1),\Pi''(2),...,\Pi''(j),\Pi''(j+1),...,\Pi''(n))$,
gdzie punkt krzyżowania j jest określony na drodze przeszukiwania:
$j = \arg\min_{1 \le k \le n} \{ \min\{ f(\Pi'(1),...,\Pi'(k),\Pi''(k+1),...\Pi''(n)),f(\Pi''(1),...,\Pi''(k),\Pi'(k+1),...\Pi'(n)) \} \}$.

4. Operator mutacji z przeszukiwaniem.

Operator 4 wyznacza jednego potomka $\Pi^1=(\Pi(1),...,\Pi'(j),...,\Pi(n))$ na podstawie jednego rodzica $\Pi=(\Pi(1),...,\Pi(j),...,\Pi(n))$:

(a) Wybierz liczbę naturalną j, $1 \le j \le n$.

(b) Określ element $\Pi'(j)$ na drodze przeszukiwania:
$(\Pi(1),...,\Pi'(j),...,\Pi(n)) = \arg\min \{ f((\Pi(1),...,\Pi''(j),...,\Pi(n))): \Pi''(j) \in D(j), \Pi''(j) \ne \Pi(j) \}$

5. Jednopunktowy operator krzyżowania z poprawą poziomu efektywności w punkcie krzyżowania.

Operator 5 wyznacza dwa potomki:
$\Pi^1=(\Pi'(1),\Pi'(2),...,\Pi'(j-1),\Pi^1(j),\Pi''(j+1),...,\Pi''(n))$,
$\Pi^2=(\Pi''(1),\Pi''(2),...,\Pi''(j-1),\Pi^2(j),\Pi'(j+1),...,\Pi'(n))$,
na podstawie dwóch rodziców:
$\Pi' =(\Pi'(1),\Pi'(2),...,\Pi'(j-1),\Pi'(j),\Pi'(j+1),...,\Pi'(n))$,
$\Pi''=(\Pi''(1),\Pi''(2),...,\Pi''(j-1),\Pi''(j),\Pi''(j+1),...,\Pi''(n))$,
gdzie punkt krzyżowania jest określony na drodze losowania, $\Pi^1(j)=(j',k^1)$, $\Pi^2(j)=(j'',k^2)$, $\Pi'(j)=(j',k')$, $\Pi''(j)=(j'',k'')$ oraz wartości k^1, k^2 są poziomami, które minimalizują, odpowiednio, funkcje $f(\Pi^1)$, $f(\Pi^2)$.

6. Dwupunktowy operator krzyżowania z przeszukiwaniem.

Operator 6 wyznacza dwa potomki Π^1, Π^2 na podstawie dwóch rodziców:
$\Pi' = (\Pi'(1),...,\Pi'(n))$, $\Pi'' = (\Pi''(1),...,\Pi''(n))$ za pomocą instrukcji:

(a) Wybierz liczbę naturalną (jeden punkt krzyżowania) i, $1 \le i \le n$. Jeżeli $i < n/2$, to przejdź do instrukcji (b), w przeciwnym przypadku przejdź do instrukcji (c).

(b) Drugi punkt krzyżowania ustal następująco:

$j = \arg \min_{i+1 \le k \le n-1} \{\min \{ f(\Pi'(1),...,\Pi'(i),\Pi''(i+1),...,\Pi''(k),\Pi'(k+1),...,$
$\Pi'(n)),f(\Pi''(1),...,\Pi''(i),\Pi'(i+1),...,\Pi'(k),\Pi''(k+1),...,\Pi''(n)) \} \}$.

Potomki mają postać:

$\Pi^1 = (\Pi'(1),...,\Pi'(i),\Pi''(i+1),...,\Pi''(j),\Pi'(j+1),...,\Pi'(n))$,

$\Pi^2 = (\Pi''(1),...,\Pi''(i),\Pi'(i+1),...,\Pi'(j),\Pi''(j+1),...,\Pi''(n))$.

(c) Instrukcja podobna do (b) z tym, że drugi punkt krzyżowania jest poszukiwany w przedziale $1 \le j \le i-1$.

Uwaga: występująca w opisach operatorów genetycznych czynność "wybierz" jest realizowana za pomocą mechanizmu losowania z rozkładem równomiernym.

4. Komputerowy system optymalizacji

Opisany powyżej algorytm genetyczny, dla zagadnienia MGAP, został zakodowany w języku PASCAL 6.0 na komputery zgodne z IBM PC pracujące pod systemem operacyjnym MS DOS. Po wykonaniu dużej liczby eksperymentów obliczeniowych zauważono celowość zmian wagi α funkcji kary $V(\Pi)$ dla rozwiązań Π. Ostatecznie w zrealizowanej wersji systemu komputerowego GA1_MGAP przyjęto następującą zasadę ustalania, w zależności od przebiegu procesu optymalizacji, wartości α. Na początku procesu optymalizacji, kiedy w wylosowanych rozwiązaniach naruszenie ograniczeń (3) jest znaczne, wartość $\alpha=\alpha_1$. Zmiana wartości α z α_1 na α_2 następuje po wyznaczeniu pierwszego, trafiającego do zbioru POPULACJA, potomka nie naruszającego ograniczeń (3). W dalszym ciągu procesu optymalizacji α przyjmuje wartość α_2, jeżeli $V(\Pi_{min})>0$ i α_3 jeżeli $V(\Pi_{min})=0$, gdzie $\alpha_1>\alpha_2>\alpha_3$, $\Pi_{min} = \min \{ f(\Pi): \Pi \in \text{POPULACJA}\}$ jest najlepszym rozwiązaniem w zbiorze POPULACJA.

Tak więc, po początkowym okresie optymalizacji, raczej z dużą wartością $\alpha = \alpha_1$ potrzebną do szybkiego zmniejszenia wartości $V(\Pi)$ generowanych potomków, waga α oscyluje między dwiema wartościami α_2 oraz α_3. Podejście to sprzyja generowaniu potomków z obszarów granicznych pomiędzy zbiorami rozwiązań niedopuszczalnych i dopuszczalnych. Zauważyliśmy decydujący wpływ zestawu wag $(\alpha_1, \alpha_2, \alpha_3)$ na efektywność procesu optymalizacji. W obecnej chwili trudno wskazać na inną metodę doboru wartości tych wag dla konkretnego zagadnienia niż metoda prób.

Przyjęto także, że liczba iteracji algorytmu jest równa liczbie wywołań operatorów genetycznych podczas procesu optymalizacji.

5. Przykład

Podamy teraz wyniki uzyskane za pomocą programu GA1_MGAP podczas rozwiązywania zadania testowego zamieszczonego w pracy [6], gdzie n=30, m=7 i l=3. Najlepsze rozwiązanie obliczone dla tego testu w czasie 120,07 sekund CPU stacji DEC 5000/120 za pomocą algorytmu typu "tabu search" specjalnie zaprojektowanego dla zagadnienia MGAP ma wartość 691634. W tej samej pracy znajduje się informacja o próbie znalezienia rozwiązania optymalnego tego zadania testowego za pomocą strategii "najpierw-w głąb" na Cplex(MIP) w ciągu 50 godzin. W tym czasie komputer nie znalazł nawet jednego rozwiązania dopuszczalnego.

Z pomocą systemu GA1_MGAP, na komputerze IBM PC 486DX2, dla parametrów: M=150, α_1 =2000, α_2=500, α_3=192, p_1=0.05, $p_2,...,p_6$=0.19 po wykonaniu 80 tys. iteracji uzyskano najlepsze rozwiązanie o wartości 690624. Po 120 tys. iteracji (6800sek.) w zbiorze POPULACJA znajdowało się 96 różnych rozwiązań, o 15 wartościach, od 690624 do 691632, lepszych od znalezionego w [6]; najgorsze rozwiązanie w tym zbiorze miało wartość 692767. Cztery najlepsze rozwiązania o V(Π)=0, z(Π)=690624 mają postać:

Π_1 = ((5,2),(2,2),(7,1),(1,1),(1,1),(1,2),(7,1),(4,1),(4,1),(4,1),
(5,1),(7,2),(3,2),(6,2),(7,1),(1,1),(4,2),(6,2),(2,2),(4,1),
(5,2),(6,2),(6,1),(3,1),(1,2)(4,2),(7,2),(3,2),(2,2),(1,1))

Π_2 = ((5,2),(2,2),(1,1),(1,1),(1,1),(1,2),(7,1),(4,1),(4,1),(4,1),
(5,1),(7,2),(3,2),(6,2),(7,1),(7,1),(4,2),(6,2),(2,2),(4,1),
(5,2),(6,2),(6,1),(3,1),(1,2)(4,2),(7,2),(3,2),(2,2),(1,1))

Π_3 = ((5,2),(2,2),(7,1),(1,1),(1,1),(1,2),(7,1),(4,1),(4,1),(1,1),
(5,1),(7,2),(3,2),(6,2),(7,1),(1,1),(4,2),(6,2),(2,2),(4,1),
(5,2),(6,2),(6,1),(3,1),(1,2)(4,2),(7,2),(3,2),(2,2),(4,1))

Π_4 = ((5,2),(2,2),(1,1),(1,1),(1,1),(1,2),(7,1),(4,1),(4,1),(1,1),
(5,1),(7,2),(3,2),(6,2),(7,1),(7,1),(4,2),(6,2),(2,2),(4,1),
(5,2),(6,2),(6,1),(3,1),(1,2)(4,2),(7,2),(3,2),(2,2),(4,1))

LITERATURA

[1] Chmiel W., Wala K.:Dynamika algorytmu genetycznego na przykładzie problemu QAP. Zeszyty Naukowe Polit. Śląskiej, Automatyka, z.109, 29-38,1992.

[2] Glover F., Hultz J., Klingman D.:Improved computer-based planning techniques - Part II. Interfaces, 9:4,12-20,1979.

[3] Goldberg D.E.:Genetic algorithms in search, optimization and machine learning. Addison-Wesley Pub.Company, Inc.,N.Y.,1988.

[4] Holland J.:Adaptation in natural and artificial systems. Univ. of Michigan Press, Ann Arbor, MI, 1975.

[5] Kadłuczka P., Wala K.:Approximate algorithms for generalized graph partitioning problem. Materiały IX Polsko-Włoskiej i VI Polsko-Fińskiej Konferencji nt."Systems Analisis and Decision Support in Economics and Technology, Radziejowice, październik 1993 (w druku).

[6] Laguna M., Kelly J.P., Gonzalez-Velarde J.L., Glover F.: Tabu search for the multilevel generalized assignment problem. Graduate School of Business and Administration, University of Colorado at Boulder, 1991.

[7] Martello S., Toth P.:Linear assignment problems. Annals of Discrete Mathematics 31, 259- 282, 1987.

[8] Michalewicz Z.:Genetic algorithms + data structures = evolution programs. Springer-Verlag, Berlin, 1992.

[9] Wala K., Chmiel W.:Local search and genetic heuristics for the permutational optimization problems. Materiały 7. Sympozjum Międzynarodowego "System-Modelling-Control", t.2, 252-257, Zakopane, maj 1993.

Recenzent: Prof. dr hab. inż. Roman Słowiński

Wpłynęło do Redakcji do 30.04.1994 r.

Abstract

This paper considers a mathematical model of the multilevel generalized assignment problem. In general, this problem consists in assigning n tasks to m agents with a maximum of l efficiency levels. In the application presented by Glover, Hultz and Klingman, the objective is to minimize the combined a cost of production and inventory holding by determining optimal product lot sizing and optimal assignment of production to machines. There are a maximum of l possible lot sizes, and the machines work in parallel at different rates and operational costs. For a particular product-machine pair, a small lot size results in a large combined cost and vice versa. An approximate algorithm, for this problem, developed on the base of genetic algorithm is presented. The approximate algorithm exploits six genetic operators where most of them realized, exept random selection operation, simple search operation.