ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2008
Seria: AUTOMATYKA z. 150 Nr kol. 1796

Tomasz PRIMKE, Zdzisław DUDA
Politechnika Śląska

CAŁKOWITOLICZBOWY ALGORYTM EWOLUCYJNY DLA PROBLEMU BALANSOWANIA LINII MONTAŻOWEJ

Streszczenie. W pracy opisano algorytm ewolucyjny dla problemu balansowania linii montażowej. W algorytmie wykorzystano kodowanie całkowitoliczbowe. Wyniki działania algorytmu porównano z wynikami uzyskanymi za pomocą znanych metod heurystycznych.

INTEGER EVOLUTIONARY ALGORITHM FOR ASSEMBLY LINE BALANCING PROBLEM

Summary. In this paper an evolutionary algorithm is proposed for assembly line balancing problem. The EA uses integer encoding. Solutions obtained with the EA are compared with solutions obtained with known heuristic methods.

1. Wprowadzenie

Znanych jest wiele klas problemów balansowania linii montażowej [3], jednakże w zdecydowanej większości obszernej literatury dostępnej na ten temat uwaga jest skupiona na pewnej specyficznej klasie problemów zwanych SALBP (ang. Simple Assembly Line Balancing Problem). Istnieją różne wersje tego problemu. W pracy uwagę skupiono na problemie zwanym SALBP-1, w którym dla danego czasu cyklu należy zminimalizować liczbę stacji roboczych, na których będą realizowane operacje procesu zgodnie z danymi relacjami kolejnościowymi.

Zaproponowano rozwiązanie problemu SALBP-1 za pomocą całkowitoliczbowego algorytmu ewolucyjnego (CAE). Istotną rolę stanowi w nim oryginalna procedura dekodująca, która zostanie opisana w dalszej części pracy. Wyniki działania proponowanego algorytmu porównano z wynikami uzyskanymi za pomocą znanych metod heurystycznych.

2. Całkowitoliczbowy algorytm ewolucyjny (CAE)

Algorytmy ewolucyjne są algorytmami stochastycznymi, których działanie oparte jest na zasadach procesów ewolucji obserwowanych w przyrodzie. Algorytmy te przetwarzają jednocześnie wiele rozwiązań, posługując się pewnym sposobem zapisu rozwiązania, zwanym kodowaniem.

W CAE zaproponowano pewną odmianę kodowania *całkowitoliczbowego*. Chromosomy osobników w populacji są reprezentowane przez wektory nieujemnych liczb całkowitych. Elementy tych wektorów mogą być nazwane genami. Zarówno rozmiar chromosomu, jak i zbiór możliwych wartości genów mogą być potraktowane jako parametry proponowanego algorytmu; będą one uzależnione od rozwiązywanego zadania.

Pomysł z zastosowaniem kodowania całkowitoliczbowego nie jest nowy. W literaturze opisanych zostało wiele algorytmów ewolucyjnych, w których chromosomy były traktowane jako wektory liczb całkowitych o pewnym określonym rozmiarze, a poszczególne geny mogły przybierać wartości z pewnego określonego zbioru [5]. Wektorom tym przypisywano na ogół interpretację związaną bezpośrednio z konkretnym problemem. Kodowania te okażą się więc zupełnie nieprzydatne w przypadku próby rozwiązania problemów innych klas.

Kodowanie całkowitoliczbowe użyte w CAE jest inne. Wartości poszczególnych genów w chromosomie nie są ściśle związane z rozwiązywanym problemem i każdy gen może przyjmować wartości z przedziału [0; R], przy czym liczba naturalna R zwana jest rozmiarem alfabetu.

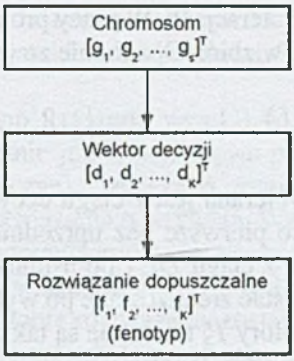
Przyjęcie całkowitoliczbowego kodowania w CAE umożliwiło zastosowanie operatorów genetycznych, które są zdolne do wykonywania operacji na wektorach liczb całkowitych [5]. Operatory te można zastosować we *wszystkich* aplikacjach CAE, niezależnie od klasy rozwiązywanego problemu. Jest to kolejna cecha omawianego algorytmu, która wyraźnie odróżnia go od innych algorytmów ewolucyjnych opisywanych w literaturze.

Każdy chromosom jest przekształcany na fenotyp. Sposób tego przekształcenia jest ściśle związany zarówno z kodowaniem, jak i z rozwiązywanym problemem. Procedura dekodująca w CAE (poza wyznaczaniem wartości funkcji przystosowania dla danego fenotypu) jest *jedynym* miejscem, w którym uwzględniona jest specyfika danego problemu. Z punktu widzenia operatorów genetycznych, jak również procedury generującej populację początkową, ważna jest tylko liczba genów oraz ich wartości. W procedurze dekodującej wartości genów są *interpretowane* w zależności od klasy rozwiązywanego problemu.

Dostosowanie CAE do rozwiązywania problemu danej klasy polega więc na zaproponowaniu odpowiedniej procedury dekodującej, zwanej dekoderem. W procedurze dekodującej można uwzględnić wszystkie ograniczenia związane z danym problemem, zastosować algorytmy optymalizacji lokalnej, zmodyfikować chromosom itp.

3. Zastosowanie CAE do problemu SALBP-1

W problemie SALBP-1 danych jest K operacji do wykonania na stacjach roboczych. Dla operacji tych określone są relacje kolejnościowe. Dany jest również cykl C, czyli okres czasu, przez jaki operacje mogą być wykonywane na jednej stacji roboczej. Rozwiązaniem problemu SALBP-1 jest takie przyporządkowanie operacji do stacji roboczych, aby były zachowane relacje kolejnościowe, czas cyklu nie był przekroczony na żadnej ze stacji, a liczba stacji N była jak najmniejsza. Jako dodatkowe kryteria przyjęto też dwa wskaźniki: efektywność linii (ang. *line efficiency*) LE oraz współczynnik gładkości (ang. *smoothness index*) SI. Są one zdefiniowane następująco:



Rys. 1. Dwuetapowa procedura dekodująca

$$LE = \frac{\sum_{i=1}^{N} T_i}{N \cdot C} \cdot 100\% \tag{1}$$

$$SI = \sqrt{\sum_{i=1}^{N} (T_{MAX} - T_i)^2} \tag{2}$$

gdzie:

C - cykl,

N - liczba stacji roboczych,

T_i - czas wykonywania operacji na i-tej stacji roboczej,

$T_{MAX} = \max_i T_i$.

Rozwiązanie problemu z zadanym cyklem można wyznaczyć na podstawie dopuszczalnej permutacji operacji. Znając taką permutację, można wyznaczyć przyporządkowanie operacji do stacji roboczych, przydzielając kolejne operacje do stacji tak długo, jak długo nie zostanie przekroczony zadany czas cyklu. Po przydzieleniu operacji do pierwszej stacji przydziela się następne operacje do drugiej stacji itd.

Ponieważ CAE przetwarza chromosomy dane w postaci wektorów nieujemnych liczb całkowitych, więc w procedurze dekodującej należy zaproponować metodę przekształcenia tych wektorów w fenotypy będące permutacjami dopuszczalnymi operacji.

Dla problemu SALBP-1 zaproponowano dwuetapową procedurę dekodującą, której schemat jest przedstawiony na rys. 1. Procedura ta najpierw przekształca chromosom na wektor decyzji dopuszczalnych, będący permutacją dopuszczalną operacji. Następnie, przez przypisanie kolejnych operacji ze znanej permutacji dopuszczalnej do kolejnych stacji roboczych, wyznaczany jest fenotyp.

Proponuje się iteracyjny sposób konstrukcji fenotypu $[f_1, f_2, \ldots, f_K]^T$ dla chromosomu $[g_1, g_2, \ldots, g_s]^T$ zgodnie ze wzorem (3):

$$P_i = P_{i-1} \cup \{d_i\}, \qquad P_0 = \emptyset \tag{3}$$

gdzie: P_i, i=1,2,...,K, jest uporządkowanym zbiorem operacji, a d_i jest decyzją wybraną z ciągu decyzji $D_i = (c_{1,i}, c_{2,i}, \ldots, c_{L,i})$ o wartości $d_i = c_{j,i}$, $j \in \{1, 2, \ldots, L\}$. Liczba możliwych do podjęcia decyzji L w ogólnym przypadku zależy od danego zadania (liczby operacji K oraz relacji kolejnościowych pomiędzy operacjami) oraz od decyzji

podejmowanych w poprzednich iteracjach. W pracy proponuje się odwzorowanie zbioru wartości genów z chromosomu w zbiór D_i zgodnie ze wzorem (4)

$$j = (g_i \bmod L) + 1, \qquad i = \{1, 2, ..., s\}, \quad L > 1 \tag{4}$$

Dla $L = 1$, $d_i = c_{1,i}$.

Pierwsza decyzja d_1 wybierana jest z ciągu decyzji D_1 zawierającego te operacje, które można wykonać jako pierwsze bez uprzedniego wykonywania innych operacji. Ciąg decyzji D_i powstaje z ciągu D_{i-1} po usunięciu z niego d_{i-1} i uzupełnieniu tymi operacjami, które mogą zostać zrealizowane po wykonaniu operacji ze zbioru P_{i-1} (zgodnie z grafem operacji). Zbiory P_i tworzone są tak długo, aż powstanie permutacja dopuszczalna P_K (czyli poszukiwany fenotyp).

Przy konstrukcji dopuszczalnej permutacji P_K należy $K - 1$ razy skorzystać ze wzoru (4). Chromosom zatem powinien zawierać $s = K - 1$ genów.

Ciąg D_i może się składać z co najwyżej tylu elementów, ile jest operacji w problemie SALBP-1. Aby w przypadku takiego ciągu indeks j we wzorze (4) mógł być równy K (liczbie operacji), wartość genu g_i musi być odpowiednio duża. Gdyby rozmiar alfabetu R był mniejszy od L-1, to indeks j nigdy by nie mógł być równy K. Widać więc, że dla problemu z K operacjami rozmiar alfabetu R powinien być równy przynajmniej K.

4. Badania numeryczne

Aby sprawdzić działanie algorytmu opisanego w poprzednim punkcie, przeprowadzono badania numeryczne. Rozwiązywano zadania SALBP-1, których dane są dostępne na stronie internetowej [3].

Do porównania użyto wybranych metod heurystycznych znanych z literatury:

- metoda Killbridge'a i Westera [4] (jako pierwsze wykonywane są te operacje, które mają mniejszą liczbę poprzedników),
- metoda Hoffmanna [2] (bazująca na macierzy relacji kolejnościowych),
- IUFF-NOF, IUFF-NOIF, IUFF-NOP, IUFF-WET oraz IUFF-BRPW [1].

W badaniach numerycznych przyjęto, że przy rozwiązywaniu danego problemu algorytm ewolucyjny zostanie wykonany 20 razy, a za wynik ostateczny uznane zostanie najlepsze spośród 20 rozwiązań. Stosowanym kryterium stopu, w jednym wykonaniu (realizacji) algorytmu, była realizacja 900 pokoleń.

Dla każdego rozwiązania uzyskanego za pomocą algorytmu CAE były również wyznaczane dodatkowe wskaźniki. Przede wszystkim analizowano odchylenie standardowe σ uzyskiwanych wyników (wyznaczone na podstawie dwudziestu rozwiązań). Mała wartość σ mogła świadczyć o dużej powtarzalności algorytmu. Innym ważnym wskaźnikiem była średnia liczba pokoleń G_{sr}, po których było znajdywane najlepsze rozwiązanie; mała wartość tej liczby świadczy o tym, że algorytm szybko znajduje rozwiązanie zadania. Pewną miarą jakości algorytmu może być też standardowe odchylenie średniej liczby pokoleń σ_G, którego mała wartość świadczy o tym, że jako kryterium stopu można przyjąć liczbę pokoleń zbliżoną do G_{sr}.

W badaniach rozważano dwa zadania: Bowmana i Gunthera [3]. Zadania te różnią się liczbą operacji oraz relacjami kolejnościowymi. W zadaniu Bowmana jest 8 operacji, a w zadaniu Gunthera 35.

W tabeli 1 przedstawiono uzyskane wyniki dla zadania Bowmana. Algorytm ewolucyjny znajdywał rozwiązanie już w pierwszym pokoleniu (z tego powodu w tabelach pominięto oceny statystyczne). Oznacza to, że mechanizmy ewolucyjne nie były w ogóle wykorzystywane do rozwiązania tych zadań. Wystarczyło użycie samego dekodera i wygenerowanie losowo odpowiednio dużej liczby chromosomów. Powodem tego była bardzo mała przestrzeń rozwiązań dopuszczalnych dla tego zadania, w związku z czym prawdopodobieństwo, że dobre rozwiązanie zostanie wygenerowane losowo, było stosunkowo duże. Badane proste metody heurystyczne również dobrze poradziły sobie z tym zadaniem.

Tabela 1

Wyniki dla zadania Bowmana

	cykl = 18			cykl = 25			cykl = 35		
	L. stacji	LE	SI	L. stacji	LE	SI	L. stacji	LE	SI
K&W	5	83%	8.66	4	75%	16.40	3	71%	23.02
Hoffmann	5	83%	8.66	4	75%	16.40	3	71%	23.02
IUFF-WET	5	83%	6.63	4	75%	14.46	3	71%	23.02
IUFF-NOP	5	83%	8.66	4	75%	16.40	3	71%	18.97
IUFF-NOIF	5	83%	8.66	4	75%	16.40	3	71%	23.02
IUFF-NOF	5	83%	8.66	4	75%	16.40	3	71%	23.02
IUFF-BRPW	5	83%	8.66	4	75%	16.40	3	71%	23.02
CAE	5	83%	6.63	4	75%	11.09	3	71%	11.05

W tabelach 2 i 3 pokazano wyniki uzyskane dla zadania Gunthera. Zadanie to było już znacznie trudniejsze. Wyniki uzyskiwane przez metody heurystyczne były już bardziej zróżnicowane, przy czym wyraźnie wyróżniała się metoda IUFF-NOIF. Algorytm CAE dał jednak równie dobre wyniki. Warto zauważyć, że wyniki te były bardzo zbieżne (zerowa wartość odchylenia standardowego). Najlepsze rozwiązanie było uzyskiwane stosunkowo szybko (po małej liczbie pokoleń) w przypadku małych i dużych wartości cyklu oraz wyraźnie później w przypadku średniego cyklu. Może to mieć związek z rozmiarem przestrzeni rozwiązań dopuszczalnych, która zależy od czasu cyklu.

5. Podsumowanie

Proponowany algorytm CAE cechuje się stosunkowo prostym dekoderem. Przeprowadzone badania numeryczne miały na celu porównanie tego algorytmu z prostymi metodami heurystycznymi. Wyniki przedstawione w tabelach 1, 2 i 3 wykazały, że algorytm CAE daje wyniki porównywalne do wyników uzyskiwanych przez proste metody heurystyczne.

Algorytm CAE, jakkolwiek dawał dobre wyniki we wszystkich sprawdzanych przypadkach, to jednak działał wolniej w porównaniu z prostymi metodami heurystycznymi. Biorąc jednak pod uwagę fakt, że problem SALBP-1 dotyczy określenia struktury przedsiębiorstwa produkcyjnego (albo ściślej: konkretnego fragmentu takiego przedsiębiorstwa, pewnego podsystemu produkcyjnego), łatwo jest zauważyć, że problem taki

Tabela 2

Wyniki dla zadania Gunthera

	cykl = 42			cykl = 65			cykl = 90		
	L. stacji	LE	SI	L. stacji	LE	SI	L. stacji	LE	SI
K&W	14	82%	43.37	9	82%	41.76	7	76%	93.90
Hoffmann	15	76%	55.49	9	82%	43.22	6	89%	30.58
IUFF-WET	14	82%	45.13	9	82%	41.47	6	89%	34.10
IUFF-NOP	14	82%	35.31	9	82%	44.99	6	89%	28.79
IUFF-NOIF	13	88%	24.64	8	92%	22.65	6	89%	46.38
IUFF-NOF	15	76%	53.88	9	82%	48.69	6	89%	29.10
IUFF-BRPW	15	76%	53.94	9	82%	46.58	6	89%	30.38
CAE	13	88%	21.98	8	92%	9.85	6	89%	9.54

Tabela 3

Wyniki dla zadania Gunthera

	cykl = 42	cykl = 65	cykl = 90
	L. stacji	L. stacji	L. stacji
CAE	13	8	6
σ	0.00	0.00	0.00
G_{sr}	3.05	110.65	7.15
σ_G	1.70	160.53	7.31

jest rozwiązywany najczęściej raz na kilka miesięcy (w szczególnym przypadku: tygodni). Zmiana struktury linii montażowej (produkcyjnej) wiąże się z ogromnymi kosztami. Zadanie SALBP-1 nie musi być więc rozwiązywane bardzo szybko i bardzo często; nawet czas rzędu godzin jest akceptowalny. Dlatego też umiejętność znalezienia rozwiązania problemu SALBP-1 w czasie rzędu minut nie jest wadą danego algorytmu.

Biorąc pod uwagę wszystkie powyższe uwagi, można stwierdzić, że proponowany algorytm CAE jest efektywnym narzędziem do rozwiązywania problemu SALBP-1.

Praca finansowana w ramach projektu badawczego KBN nr 3T11A02229.

BIBLIOGRAFIA

1. Hackman S. T., Magazine M. J., Wee T. S.: Fast Effective Algorithms for Simple Assembly Line Balancing Problems. Operations Research, 1989, vol. 37(6), p. 916-924.
2. Hoffmann T. R.: Assembly Line Balancing with a Precedence Matrix. Management Science, 1963, vol. 9(4), p. 551-562.
3. Homepage for assembly line optimization research, http://www.wiwi.uni-jena.de/Entscheidung/alb/index.htm
4. Kilbridge M. D., Wester L.: A Heuristic Method of Assembly Line Balancing. Journal of Industrial Engineering, 1961, vol. 12(4), p. 292-298.

5. Pawlak M.: Algorytmy ewolucyjne jako narzędzie harmonogramowania produkcji. Wydawnictwo Naukowe PWN, Warszawa 1999.
6. Primke T.: Analiza efektywności algorytmu ewolucyjnego w wybranych problemach sterowania dyskretnymi procesami produkcji. Rozprawa doktorska, Instytut Automatyki, Politechnika Śląska, Gliwice 2008.

Recenzent: Prof. dr hab. inż. Zbigniew Banaszak

Abstract

In this paper an evolutionary algorithm is proposed for assembly line balancing problem. The EA uses integer encoding and classical genetic operators. A two-stage decoding procedure is proposed. The fitness function is based on measures for the assembly line balancing problem known from literature. Solutions obtained with the EA are compared with solutions obtained with known heuristic methods.