AUTOMATYKA • 2005 • Tom 9 • Zeszyt 1–2

Jarosław Pempera*, Czesław Smutnicki*

# Minimalizacja czasu cyklu wytwarzania na linii. Podejście genetyczne z ekspresją genów**

## 1. Wprowadzenie

Wytwarzanie cykliczne jest podstawową strategią działania systemów produkcyjnych realizujących wytwarzanie wieloasortymentowe, wielkoseryjne przy wolnozmiennym w czasie asortymencie wyrobów. System cykliczny dostarcza na wyjście, co pewien kwant czasu (czas cyklu), ustaloną mieszankę produktów lub półproduktów. Skład ilościowy mieszanki zależy od średnio- i długoterminowych zamówień klientów. Minimalizując czas cyklu, dla ustalonego zestawu zadań w cyklu zwiększamy wydajność systemu oraz stopień wykorzystania maszyn. Poprawę efektywności można także uzyskać, ograniczając magazynowanie półproduktów w magazynach pośrednich. Prowadzi to do ograniczeń typu *no store* (zakaz składowania międzystanowiskowego) lub *limited store* (ograniczony obszar składowania międzystanowiskowego, tzw. bufor). W wytwarzaniu cyklicznym wprowadzenie ograniczeń tego typu jest naturalną koniecznością.

Deterministyczne problemy cykliczne z ograniczeniami składowania należą do jednych z najtrudniejszych zagadnień optymalizacji dyskretnej. Świadczy o tym stosunkowo niewielka liczba pozycji literaturowych oraz kłopotliwe numerycznie procedury wyznaczania długości czasu cyklu i/lub harmonogramu. Co więcej, analiza wariantów cyklicznych nie wyszła poza podstawowe modele szeregowania. Dla wielu praktycznych problemów, jak np. gniazdowy problem szeregowania (*job-shop problem*), pojęcie czasu cyklu nie zostało do tej pory zdefiniowane.

Silna *NP*-trudność, już w wielu najprostszych badanych wersjach problemu, ogranicza zakres stosowania algorytmów dokładnych do instancji o małej liczbie zadań. Z tego powodu, do wyznaczania satysfakcjonujących rozwiązań stosuje się powszechnie szybkie algorytmy przybliżone oparte na technikach przeszukiwań lokalnych. Prawie wszystkie metody tego typu opierają się na dwupoziomowej dekompozycji problemu:

1) wyznaczenie optymalnej kolejności zadań w cyklu (poziom górny),
2) wyznaczenie minimalnego czasu cyklu dla danej kolejności zadań (poziom dolny).

* Instytut Informatyki Automatyki i Robotyki Politechniki Wrocławskiej
** Praca była finansowana ze środków KBN w latach 2003–2006, jako projekt badawczy nr T11A01624

Rozwiązanie (wielokrotne) zagadnienia dolnego poziomu jest stosunkowo czasochłonne, ogólnie wymaga rozwiązania pewnego zagadnienia PL, w szczególnych przypadkach może być sprowadzone do problemu minimalnego przepływu lub specyficznej drogi w grafie. Co więcej, brak jest dedykowanych własności problemu, pozwalających na ograniczenie liczności lokalnie przeglądanego sąsiedztwa i/lub przyspieszenie szybkości jego przeglądania. Dlatego z szerokiej gamy metod poszukiwań, odrzuciliśmy początkowo technikę przeszukiwania z zabronieniami – nadmiernie kosztowną w omawianym przypadku. Jednym z rozsądniejszych podejść wydaje się metoda genetycznego i/lub symulowanego wyżarzania. W tej pracy dyskutujemy pierwszy z wymienionych algorytmów.

W literaturze można spotkać wiele algorytmów genetycznych dla problemów szeregowania zadań np. [1]. Powszechnie znane są ich zalety i wady. W większości przypadków uzyskanie najlepszego efektu wymaga czasochłonnego „dostrojenia" algorytmu do klasy problemu i/lub klasy przykładów testowych. Co więcej, uniwersalne metody tworzenia kodowania rozwiązań za pomocą ciągów binarnych okazały się mało przydatne dla problemów szeregowania. Pociągnęło to za sobą pewną niejednorodność w technikach kodowania cech, zmierzając w stronę koncepcji: rozwiązanie = obiekt kombinatoryczny (np. permutacja, zestaw permutacji, podziały zbioru) = osobnik = genotyp. (Upraszczające założenie osobnik = genotyp zmniejsza liczbę parametrów strojenia przy projektowaniu algorytmu.) To z kolei przeniosło ciężar budowy algorytmu genetycznego na zaprojektowanie odpowiedniego operatora krzyżowania osobników, dedykowanego do określonych zastosowań, upatrując w nim „motor postępu" poszukiwań. Sztuczna ewolucja tworzy w ten sposób nowe osobniki znacznie różniące się od rodziców, co sugeruje kompletnie demokratyczną i wielokierunkową politykę Natury. Taki demokratyzm jest zbędny w praktyce szeregowania, gdzie wykorzystanie historii dotychczasowych poszukiwań pozwala skutecznie ukierunkować dalsze poszukiwania. Podobnie jest w Naturze, gdzie osobnik jest określany zestawem cech fenotypowych będących wynikiem ekspresji olbrzymiej liczby genów, często przenoszonych z pokolenia na pokolenia w formie uśpionej, wyrażających „pośrednio" informacje z historii rozwoju pokoleń. Prowadzi to do koncepcji rozwiązanie = osobnik = fenotyp ≠ genotyp. W pracy analizujemy nowy sposób kodowania osobników (genotyp) oraz pewien mechanizm ekspresji genów pozwalający z genotypu stworzyć osobnika (fenotyp). Efektywność zaproponowanego podejścia została przebadana eksperymentalnie.

## 2. Opis problemu

W systemie wytwórczym należy wykonać $N$, niekoniecznie różnych, zadań danych zbiorem $J = \{1, 2, ..., N\}$, przy użyciu $m$ maszyn indeksowanych 1, 2, ..., $m$. Pojedyncze zadanie odpowiada wytworzeniu jednej sztuki produktu finalnego (lub półproduktu). Wykonanie zadania odbywa się w kolejnych $m$ etapach, w jednakowy sposób dla wszystkich zadań. Etap $i$ wykonywany jest przez maszynę o numerze $i$, $i = 1, ..., m$. Ze względu na strategię wytwarzania, tj. brak możliwości składowania wyrobów podczas produkcji w magazynach i buforach międzystadialnych, zadanie zakończone na maszynie $i$, którego nie można przekazać na maszynę $i$+1 ze względu na jej zajętość, musi pozostawać na $i$ (blokować ją) aż do chwili zwolnienia $i$+1. Każde zadanie $j \in J$ można podzielić na sekwencję $m$

operacji $O_{1j}$, $O_{2j}$, ..., $O_{mj}$ wykonywanych kolejno na maszynach. Operacja $O_{ij}$ odpowiada wykonywaniu zadania $j$ na $i$-tej maszynie w czasie $p_{ij} > 0$. W danej chwili maszyna może wykonywać tylko jedno zadanie, oraz zadanie może być wykonywane tylko na jednej maszynie. Systemy z ograniczonymi pojemnościami buforów lub ich brakiem są obiektem zainteresowań wielu badaczy [2]–[5].

Przyjmujemy dalej, że zbiór $J$ składa się z $t$ różnych typów zadań o licznościach $r_1,\ldots,r_t$, odpowiednio: zadania tego samego typu mają identyczne czasy trwania poszczególnych operacji. Stąd mamy $N = \sum_{i=1}^{t} r_i$. Jeśli istnieje liczba całkowita $c > 1$, będąca największym wspólnym dzielnikiem liczb $r_1$, …, $r_t$, to zbiór $J$ możemy podzielić na $c$ identycznych podzbiorów zadań, z których każdy zawiera $n = r_1/c + \ldots + r_T/c$ zadań. Każdy taki podzbiór, nazywany MPS (*minimal part set*). Dalej będziemy się zajmować wyłącznie zadaniami w obrębie pojedynczego MPS.

W systemach cyklicznych, najpierw wykonywane są w określonej kolejności zadania należące do pierwszego MPS, następnie w tej samej kolejności wykonywane są zadania z drugiego MPS, itd. W rozważanym systemie przepływowym z ograniczeniami bez magazynowania dopuszczalna kolejność wykonywania zadań na maszynach musi być identyczna na każdej maszynie [3]. Zatem, kolejność wykonywania w MPS możemy opisać za pomocą permutacji $\pi$ elementów zbioru $\{1, \ldots, n\}$. Oczywiście kolejność realizacji wszystkich zadań z $J$ powstaje poprzez $c$-krotne złożenie $\pi$.

Harmonogram wykonywania zadań $k$-tego MPS opisujemy za pomocą macierzy $[S^k]_{m\times n}$, gdzie $S_{i,j}^k$ oznacza termin rozpoczęcia wykonywania zadania $j$ na maszynie $i$. Dopuszczalny harmonogram wyznaczony dla kolejności wykonywania zadań $\pi$ musi spełniać następujące ograniczenia:

$$S_{i,\pi(j)}^k + p_{i,\pi(j)} \le S_{i+1,\pi(j)}^k, \quad i = 1, ..., m-1, \quad j = 1, ..., n \tag{1}$$

$$S_{i,\pi(j)}^k + p_{i,\pi(j)} \le S_{i,\pi(j+1)}^k, \quad i = 1, ..., m, \quad j = 1, ..., n-1 \tag{2}$$

$$S_{i,\pi(j)}^k \le S_{i,\pi(j+1)}^k, \quad i = 1, ..., m-1, \quad j = 1, ..., n-1 \tag{3}$$

$$S_{i,\pi(n)}^k + p_{i,\pi(n)} \le S_{i,\pi(1)}^{k+1}, \quad i = 1, ..., m \tag{4}$$

$$S_{i+1,\pi(n)}^k \le S_{i,\pi(1)}^{k+1}, \quad i = 1, ..., m-1 \tag{5}$$

Ograniczenie (1) wynika z porządku technologicznego wykonywania zadań, natomiast (2) z jednostkowej przepustowości maszyn (w danej chwili może wykonywać tylko jedno zadanie). Ograniczenie (3) modeluje ograniczenie bez magazynowania i oznacza, że rozpoczęcie wykonywania zadania $\pi(j+1)$ na maszynie $i$, może rozpocząć się nie wcześniej niż w momencie rozpoczęcia zadania $\pi(j)$ na maszynie $i+1$; tj. w momencie zwolnienia przez to zadanie maszyny $i$. Ograniczenia (4) i (5) dotyczą relacji czasowych pomiędzy wykonywaniem ostatniego zadania $k$-tego MPS-a i pierwszego zadania w następnym $(k+1)$-szym MPS. Ograniczenia te są analogiczne do (2) i (3).

Założenie o cykliczności wytwarzania implikuje, że istnieje stała $T$ taka, że:

$$S^{k+1}_{i,\pi(j)} = S^{k}_{i,\pi(j)} + T, \quad i = 1, ..., m, \quad j = 1, ..., n, \quad k = 1, 2, ... \tag{6}$$

Okres $T$ nazywamy czasem cyklu. W sposób oczywisty, $T$ zależy od kolejności wykonywania zadań $\pi$. W dalszej części pracy będziemy poszukiwali (dla każdej ustalonej $\pi$) minimalnej wartości $T$, dla której spełnione są ograniczenia (1)–(6), tzw. minimalny czas cyklu, oznaczony dalej przez $T(\pi)$.

Przejdźmy zatem do postawienia problemu optymalizacyjnego. Niech $\Pi$ będzie zbiorem wszystkich permutacji określonych na zbiorze $\{1, \ldots, n\}$. Chcemy znaleźć permutację $\pi^* \in \Pi$ taką, że

$$T(\pi^*) = \min_{\pi \in \Pi} T(\pi) \tag{7}$$

gdzie $T(\pi)$ jest minimalnym czasem cyklu dla kolejności $\pi$. Pokażemy dalej jak wyznaczyć $T(\pi)$.

Niech $\pi$ będzie ustalone, zaś $T$ niech oznacza czas cyklu zgodnie z definicją (6). Podstawiając (6) do (1)–(5) otrzymamy warunki odnoszące się tylko do jednego $k$-tego MPS, co pozwala pominąć górny indeks $k$ w oznaczeniach. Stąd $T(\pi)$ oraz harmonogram $S_{ij}$, $i = 1, ..., m$, $j = 1, ..., n$ każdego cyklu można wyznaczyć poprzez kosztowne czasowo rozwiązanie zadania PL postaci:

$$T(\pi) = \min_{T, S_{ij}} T \tag{8}$$

$$S_{i,\pi(j)} + p_{i,\pi(j)} \le S_{i+1,\pi(j)}, \quad i = 1, ..., m-1, \quad j = 1, ..., n \tag{9}$$

$$S_{i,\pi(j)} + p_{i,\pi(j)} \le S_{i,\pi(j+1)}, \quad i = 1, ..., m, \quad j = 1, ..., n-1 \tag{10}$$

$$S_{i,\pi(j)} \le S_{i,\pi(j+1)}, \quad i = 1, ..., m-1, \quad j = 1, ..., n-1 \tag{11}$$

$$S_{i,\pi(n)} + p_{i,\pi(n)} \le S_{i,\pi(1)} + T, \quad i = 1, ..., m \tag{12}$$

$$S_{i+1,\pi(n)} \le S_{i,\pi(1)} + T, \quad i = 1, ..., m-1 \tag{13}$$

W pracy [6] pokazano metodę wyznaczania $T(\pi)$ oraz harmonogramu $S_{ij}$, $i = 1, ..., m$, $j = 1, ..., n$ poprzez rozwiązanie zadania minimalnego przepływu w specyficznej sieci. Biorąc pod uwagę rozmiar sieci $O(nm)$ oraz złożoność algorytmu wyznaczania przepływu, otrzymamy w tym przypadku dość kosztowną obliczeniowo metodę $O(n^3m^3)$. W następnej sekcji przestawimy model grafowy pozwalający w sposób bardziej efektywny wyznaczyć czas cyklu. Zauważmy, że z (12)–(13) wynika, iż wielkość

$$LB_T = \max_{1 \le i \le m} [\max\{S_{i,\pi(n)} + p_{i,\pi(n)}, S_{i+1,\pi(n)}\} - S_{i,\pi(1)}] \tag{14}$$

jest dolnym ograniczeniem minimalnej wartości $T$, gdzie $S_{m+1,j} = S_{m,j}$, $j = 1, ..., n$.

## 3. Model grafowy

Model grafowy budujemy na bazie prostokątnego grafu siatkowego zawierającego $m$+1 wierszy i $n$+1 kolumn. Do tego celu wprowadzamy fikcyjną maszynę o numerze $m$+1, definiujemy $p_{m+1,j} = 0$, $j = 1, \ldots, n$ oraz przyjmujemy $\pi(n+1) = \pi(1)$.

Dla danej kolejności $\pi$, budujemy graf

$$G(\pi) = (V, R \cup F(\pi))$$

ze zbiorem węzłów $V$ i zbiorem łuków $R \cup F(\pi)$, gdzie odpowiednio:

$$V = \{1, ..., m+1\} \times \{1, ..., n+1\},$$

$$R = \bigcup_{k=1}^{m} \bigcup_{j=1}^{n+1} \{((k, j), (k+1, j))\},$$

$$F(\pi) = \bigcup_{k=1}^{m} \bigcup_{j=1}^{n} \{((k+1, \pi(j)), (k, \pi(j+1))\}.$$

Węzeł $(i,j) \in V$ reprezentuje $i$-tą operację zadania $\pi(j)$. Waga węzła $(i,j) \in V$ wynosi $p_{i,\pi(j)}$. Łuki ze zbioru $R$ modelują ograniczenia (10) i (12), oraz są obciążone wagą równą zero. Łuki ze zbioru $F(\pi)$ modelują ograniczenia (11) i (13); każdy łuk $((k+1, \pi(j)), (k, \pi(j+1))) \in F(\pi)$ posiada wagę minus $p_{k+1,\pi(j)}$.

Niech $D_i$, $i = 1, \ldots, m$, będzie długością najdłuższej drogi od węzła $(i,1)$ do węzła $(i,n+1)$ (wliczając $p_{i,\pi(1)}$, ale nie wliczając $p_{i,\pi(n+1)}$) w tak zdefiniowanym grafie, zaś $D_{\max} = \max\{D_1, \ldots, D_m\}$. Drogę odpowiadającą $D_{\max}$ nazywano w pracy [6] jednokrotną ścieżką wokół cylindra.

Zachodzą następujące własności.

**Własność 1.** $LB_T = D_{\max}$.

**Własność 2.** Istnieje rozwiązanie problemu (8)–(13) takie, że $T(\pi) = LB_T$.

Własności 1 i 2 pozwalają znaleźć wartość minimalnego czasu cyklu $T(\pi)$ (bez konieczności wyznaczania dokładnego harmonogramu $S_{ij}$, $i = 1, ..., m$, $j = 1, ..., n$) poprzez wyznaczenie specyficznych dróg w grafie, co można wykonać w niekosztownym czasie $O(nm^2)$. Daje to bezsporną korzyść algorytmom poszukiwań lokalnych oceniających rozwiązania explicite poprzez ich wartość funkcji celu. Otwartym problemem jest konstrukcja algorytmu wyznaczania harmonogramu dla danego $T(\pi)$.

## 4. Algorytm genetyczny

Tradycyjny schemat algorytmów genetycznych jest znacznym uproszczeniem skomplikowanego procesu naturalnej ewolucji, gdzie krzyżowanie jest traktowane jako podstawowy mechnizm dostarczania osobników o zróżnicowanych cechach [8]. Faktycznie, krzyżowanie

pozwala na wymianę informacji genetycznej zawartej w DNA, decydującej kolejno o powstawaniu iRNA, tRNA, mRNA, enzymów oraz ostatecznie o biosyntezie określonego zestawu białek, z których zbudowany jest żywy organizm. Dopiero tak utworzony organizm jest oceniany poprzez eksponowane cechy zewnętrzne pod kątem przystosowania do środowiska (rangi w społeczeństwie). Cecha fenotypowa (zewnętrzna) jest determinowana zwykle przez wiele genów wchodzących w złożone interakcje, poprzez tzw. mechanizm ekspresji genów. Można powiedzieć, że informacja zawarta w kodzie DNA jest w pewnym sensie nadmiarowa, i może przenosić informacje dotyczące historii ewolucji danego gatunku. Przykładowo, w pracy [7], zastosowano dla problemu komiwojażera istotnie różny sposób reprezentowania genotypu i fenotypu oraz specyficzny algorytm ekspresji genów oparty na pewnej metodzie heurystycznej prowadzącej do otrzymania fenotypu.

### 4.1. Fenotyp i genotyp

Naturalną reprezentacją fenotypu osobnika w problemie rozważanym w pracy jest permutacja określająca kolejność wykonywania zadań. W celu zachowania informacji genetycznej z poprzednich pokoleń, proponujemy genotyp osobnika zakodować w postaci trzech permutacji. Pierwsza jest taka sama jak permutacja określająca fenotyp, natomiast druga i trzecia są identyczna fenotypowi odpowiednio ojca i dziadka. Oczywiście w ten sposób można zapamiętać informację genetyczną pochodzącą od jeszcze wcześniejszych pokoleń.

### 4.2. Krzyżowanie

Jednym z najefektywniejszych operatorów krzyżowania dla reprezentacji permutacyjnych jest operator krzyżowania z częściowym odwzorowaniem (PMX) [9]. W proponowanym algorytmie krzyżowanie osobników odbywa się następujący sposób: w wyniku krzyżowania pierwszych permutacji pochodzących od rodziców operatorem PMX otrzymujemy pierwszą permutację potomka, drugą otrzymujemy przez skopiowanie fenotypu ojca, natomiast trzecią przez skopiowanie drugiej permutacji genotypu ojca, co łatwo można sprawdzić, która jest identyczna fenotypowi dziadka (precyzyjniej ojca ojca). Działanie operatora PMX przedstawimy na przykładzie (rys. 1).

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| O | 14 | 5 | 2 | 10 | 11 | 8 | 1 | **16** | **4** | **15** | **19** | **13** | **9** | **7** | **18** | **3** | **12** | **17** | 6 | 20 | … |
| M | 18 | 11 | 10 | 4 | 8 | 14 | 20 | **15** | **1** | **19** | **6** | **5** | **13** | **3** | **12** | **17** | **9** | **7** | 2 | 16 | … |
| S | 14 | 18 | 2 | 10 | 11 | 8 | 4 | **15** | **1** | **19** | **6** | **5** | **13** | **3** | **12** | **17** | **9** | **7** | 16 | 20 | … |
| C | 5 | 11 | 10 | 1 | 8 | 14 | 20 | **16** | **4** | **15** | **19** | **13** | **9** | **7** | **18** | **3** | **12** | **17** | 2 | 6 | … |

Grubą czcionką oznaczono fragmenty permutacji pomiędzy punktami cięcia *a* i *b*. W przypadku osobnika *S* (*C*) podkreśleniem oznaczono pozycje, na których doszło do wymiany elementów wynikającej z ciągu odwzorowań, na pozostałych pozycjach znajdują się elementy, które stoją na takiej samej pozycji jak u ojca O.

**Rys. 1.** Ilustracja działania zmodyfikowanego operatora PMX

Niech losowo wybrane punkty cięcia wynoszą $a = 8$ i $b = 18$. Dla osobników ojca O i matki M, punkty $a$ i $b$ definiują ciąg wy-mian 16↔15, 4↔1, 15↔19, 19↔6, 13↔5, 9↔13, 7↔3, 18↔12, 3↔17, 12↔9, 17↔7. Potomek S, powstaje poprzez skopiowanie pierwszej permutacji ojca O, wymianie elementów stojących na pozycjach od $a$ do $b$ zgodnie z tym ciągiem odwzorowań. Następnie spośród elementów znajdujących się na pozostałych pozycjach wybierane są elementy konfliktowe, tzn. takie, które występują w permutacji dwukrotnie. Elementy te wymieniane są na elementy jeszcze nie występujące w permutacji zgodnie ze zdefiniowanym ciągiem wymian. Dla przykładu drugi element 5 zostaje wymieniony na 13, 13 na 9, 9 na 12 i ostatecznie 12 na 18. W podobny sposób (przez odwrócenie ról osobników) generowany jest drugi z osobników.

### 4.3. Algorytm ekspresji genów

W wyniku działania krzyżowania otrzymujemy genotyp osobnika S składający się z trzech permutacji (rys. 2), gdzie FO oznacza fenotyp ojca, natomiast FD fenotyp dziadka.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 14 | 18 | 2 | 10 | 11 | 8 | 4 | **15** | **1** | **19** | **6** | **5** | **13** | **3** | **12** | **17** | **9** | **7** | 16 | 20 |
| FO | 14 | 5 | 2 | 10 | 11 | 8 | 1 | **16** | **4** | **15** | **19** | **13** | **9** | **7** | **18** | **3** | **12** | **17** | 6 | 20 |
| FD | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |

**Rys. 2.** Potomek. Genotyp

Algorytm ekspresji genów na podstawie nadmiarowych informacji zapisanych w drugiej i trzeciej permutacji, tj. informacji pochodzącej od ojca i dziadka oraz wyniku krzyżowania, tworzy fenotyp nowego potomka w dwóch fazach. W pierwszej fazie następuje modyfikacja pozycji elementów pochodzących od ojca, których pozycja została ustalona w wyniku ciągu wymian (pozycje podkreślone). Elementy te będziemy nazywali ruchomymi. Kolejno, brana jest do analizy każda para sąsiadujących ze sobą elementów w fenotypie ojca. Niech ($a$, $b$) będzie jedną z takich par, wówczas jeżeli $a$ i $b$ są elementami ruchomymi to element $b$ przesuwany jest bezpośrednio za element $a$ w tworzonym fenotypie osobnika $S$. W naszym przykładzie w wyniku tej modyfikacji 4 zostanie przesunięte za 16 (rys. 3), gdzie FS oznacza fenotyp S.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FO | 14 | 5 | 2 | 10 | 11 | 8 | 1 | **16** | **4** | **15** | **19** | **13** | **9** | **7** | **18** | **3** | **12** | **17** | 6 | 20 |
| FS | 14 | 18 | 2 | 10 | 11 | 8 | **15** | **1** | **19** | **6** | **5** | **13** | **3** | **12** | **17** | **9** | **7** | 16 | 4 | 20 |
| | $F_1$ | $C_2$ | | $F_3$ | | | | | | $F_4$ | | | | | | | | $F_5$ | | $F_6$ |

**Rys. 3.** Potomek. Tworzenie fenotypu – faza pierwsza

Przed przystąpieniem do realizacji drugiej fazy algorytmu, fenotyp S dzielony jest na fragmenty, składające się z jak największej liczby elementów pochodzących od tego samego rodzica i wykonywanych w tej samej kolejności jak u niego. Podobnie jak w pierwszej,

w drugiej fazie brana jest do analizy każda para sąsiadujących ze sobą elementów w fenotypie dziadka (rys. 4). Niech (*a*, *b*) będzie jedną z takich par, wówczas jeżeli *a* jest ostatnim elementem w pewnym fragmencie a *b* pierwszym w innym, wówczas wszystkie elementy tego drugiego wstawiane są za element *a* zachowując kolejność w jakiej występowały w tym fragmencie. W analizowanym przykładzie w wyniku działania drugiej fazy fragment $F_4$ zostanie przesunięty bezpośrednio za ostatni element fragmentu $F_1$.

| FD | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FS | 14 | **15** | **1** | **19** | **6** | **5** | **13** | **3** | **12** | **17** | **9** | **7** | 18 | 2 | 10 | 11 | 8 | 16 | 4 | 20 |

$F_1$ $F_4$

**Rys. 4.** Potomek. Tworzenie fenotypu – faza druga

## 5. Eksperyment komputerowy

Celem przeprowadzonego eksperymentu komputerowego było przebadanie wpływu zaproponowanej nadmiarowości kodowania oraz związanego z nim mechanizmu ekspresji genów na efektywność algorytmu genetycznego. W tym celu oprogramowano algorytm genetyczny zaproponowany w pracy [10]. W algorytmie tym w każdej iteracji w każdej z NGEN iteracji, dla każdego z nieparzystej liczby POPSIZE osobników należących do zbioru POP wyznaczana jest wartość funkcji dopasowania z następującej zależności:

$$fit(t) = \frac{\max_{j \in POP}(T(F(j))) - T(F(t))}{\sum_{k \in POP}(\max_{j \in POP}(T(F(j))) - T(F(k)))}, \quad t \in \text{POP},$$

gdzie:

$F(t)$ – fenotyp osobnika $t$,
$T$ – czas cyklu.

Dla każdego osobnika obliczane jest prawdopodobieństwo wyboru z zależności:

$$pr(t) = \frac{fit(t)}{\sum_{k \in POP} fit(k)}, \quad t \in \text{POP}.$$

Zgodnie z zasadą ruletki wyznaczane jest do celów reprodukcyjnych $\lfloor \text{POPSIZE}/2 \rfloor$ par rodziców. Każda para rodziców krzyżowana jest z prawdopodobieństwem PCROSS i\lub każdy z osobników poddawany jest mutacji z prawdopodobieństwem PMUTE. Mutacja polega na wymianie elementów stojących na dwóch losowo wybranych pozycjach.

Nowa populacja składa się z nowo powstałych potomków oraz najlepiej dopasowanego osobnika z populacji rodziców. W przypadku gdy populacja jest mało zróżnicowana, tj. jeże-

li powyżej POPFIT procent osobników jest wzajemnie do siebie dopasowanych, to populacja zastępowana jest populacją osobników wygenerowanych losowo, w identyczny sposób generowana jest populacja początkowa. Dopasowanie osobnika $i \in$ POP oznacza, że $\max_{j \in POP}(fit(j) - fit(i)) \leq \text{FITTOLER}$. Wynikiem działania algorytmu jest najlepsza permutacja wygenerowana podczas działania algorytmu, tj, permutacja, dla której czas cyklu jest najmniejszy. Podobnie jak w [10] algorytm został uruchomiony z następującymi wartościami parametrów POPSIZE=95, NGEN=1000, FITTOLER=1E-10, POPFIT 60, PCROSS 07.25, PMUTE 0.009.

Algorytm został zaprogramowany w języku C++ i testowany był na pierwszych sześciu grupach przykładów należących do zestawu zaproponowanego przez Taillarda [11]. Każda z tych grup składa się z 10 trudnych instancji o tej samej liczbie maszyn *m* i tej samej liczbie zadań *n*.

Algorytm genetyczny oprogramowano z trzema operatorami krzyżowania:

1) GA-PMX wykorzystujący jedynie operator PMX,
2) GA-EO wykorzystujący operator PMX i pierwszą fazę algorytmu ekspresji genów,
3) GA-E wykorzystujący operator krzyżowania PMX i dwufazowy algorytm ekspresji genów.

Każdy z algorytmów dla każdej instancji został uruchomiony 10 razy.

Dla każdej instancji problemu, dla każdego z testowanych algorytmów (GA-PMX, GA-EO, GA-E) oraz dla każdego z 10 uruchomień wyznaczono:

$T^*$ – najmniejszy czas cyklu uzyskany podczas badań algorytmów dla tej instancji,

$B_i^A = 100\%(T(\pi_i^A) - T^*)/T^*$ – błąd rozwiązania $\pi_i^A$ otrzymanego podczas *i*-tego uruchomienia algorytmu *A*.

Bazując na wyliczonych wartościach błędów dla każdej instancji i dla każdego algorytmu wyznaczono:

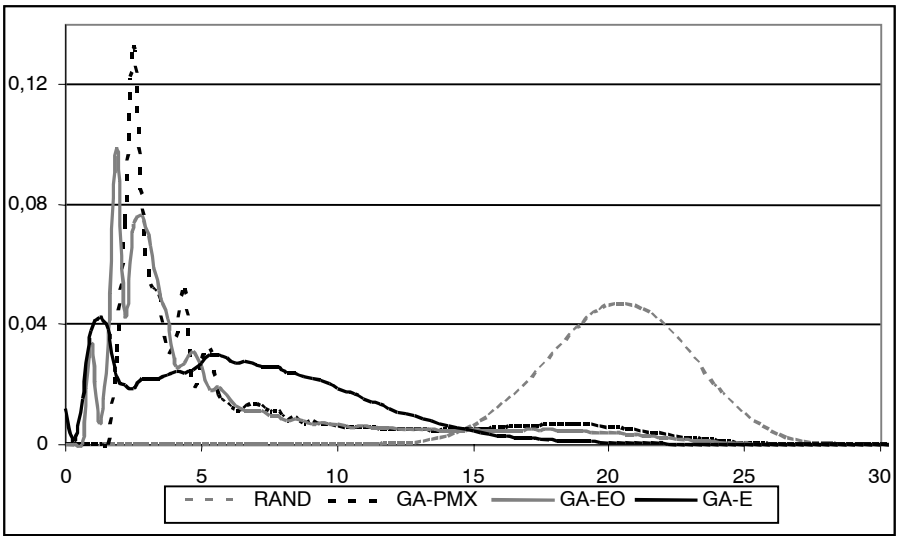
*AB* – średni błąd,
*SB* – odchylenie standardowe błędu,
*MINB* – minimalny błąd,
*MAXB* – maksymalny błąd.

Ostatecznie dla każdego algorytmu i każdej grupy przykładów wyznaczono wartości średnie wyżej wymienionych wielkości. Wyniki przedstawiono w tabeli 1.

Następnie, dla pierwszego przykładu pierwszej grupy instancji, przeprowadzono analizę przestrzeni rozwiązań przeglądanych przez badane algorytmy. W tym celu zachowano wartości błędu wyznaczonego dla każdego osobnika generowanego podczas przebiegu algorytmów badanych oraz algorytmu RAND, który generował w sposób losowy taką samą liczbę permutacji jak algorytmy genetyczne. Dla każdego algorytmu, na podstawie zbioru składającego się z 690 000 liczb wyznaczono empiryczną gęstość błędu. Wykresy empirycznych gęstości błędu zostały przedstawione na rysunku 5.

**Tabela 1**
Porównanie efektywności operatorów krzyżowania

| Grupa | GA-PMX | | | | GA-EO | | | | GA-E | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AB | SB | MINB | MAXB | AB | SB | MINB | MAXB | AB | SB | MINB | MAXB |
| 20×5 | 2,95 | 0,29 | 1,29 | 4,41 | 2,01 | 0,26 | 0,56 | 3,27 | 1,12 | 0,25 | 0,01 | 2,57 |
| 20×10 | 2,48 | 0,33 | 0,82 | 4,13 | 2,18 | 0,26 | 0,95 | 3,56 | 1,49 | 0,26 | 0,13 | 2,83 |
| 20×20 | 1,81 | 0,20 | 0,79 | 2,79 | 1,43 | 0,18 | 0,48 | 2,42 | 1,13 | 0,21 | 0,07 | 2,26 |
| 50×5 | 2,97 | 0,24 | 1,68 | 4,21 | 2,15 | 0,28 | 0,67 | 3,52 | 1,52 | 0,26 | 0,13 | 2,89 |
| 50×10 | 2,46 | 0,23 | 1,26 | 3,83 | 1,65 | 0,25 | 0,43 | 3,03 | 1,35 | 0,24 | 0,16 | 2,59 |
| 50×20 | 1,50 | 0,14 | 0,72 | 2,28 | 1,16 | 0,17 | 0,31 | 2,03 | 1,00 | 0,17 | 0,10 | 1,99 |
| Średnio | 2,36 | 0,24 | 1,09 | 3,61 | 1,76 | 0,23 | 0,57 | 2,97 | 1,27 | 0,23 | 0,10 | 2,52 |




**Rys. 5.** Wykresy empirycznych gęstości błędu

## 6. Uwagi i wnioski

Z analizy wyników prezentowanych w tabeli 1 wynika, że dla rozważanego problemu zastosowanie proponowanego nadmiarowego kodowania cech w połączeniu z proponowanym algorytmem ekspresji genów znacznie zwiększa efektywność algorytmu genetycznego.

Analizując wartości średnie błędu, możemy zauważyć, że algorytm GA-EO, który wykorzystując tylko pierwszą fazę algorytmu ekspresji genów, generuje rozwiązania o 0,6% lepsze w sensie średnim od algorytmu GA-PMX wykorzystującego klasyczny operator PMX. Dalsze zwiększenie efektywności algorytmu obserwuje się w przypadku algorytmu stosującego pełny algorytm ekspresji genów, tj. algorytmu GA-E. Algorytm ten generuje rozwiązania o przeszło 1% lepsze. Przewagę algorytmów GA-EO i GA-E nad algorytmem GA-PMX można zaobserwować we wszystkich grupach przykładów. W podobny sposób zachowują się wielkości *MINB* i *MAXB*. Analizując zachowanie się algorytmów w zależności od liczby zadań i maszyn, możemy zauważyć, że wraz ze wzrostem liczby zadań rośnie efektywność algorytmów GA-PMX i GA-EO.

## Literatura


[1] Sabena B., Iyer S.K.: *Improved genetic algorithm for the permutation flowshop scheduling problem*. Computers & Operations Research, 31, 2004, 593–606

[2] Smutnicki C.: *Some properties of scheduling problem with storage constraints*. Zeszyty Naukowe AGH Automatyka, 34, 1983, 223–232

[3] Grabowski J., Pempera J.: *Sequencing of jobs in some production system*. European Journal of Operational Research, 125, 2000, 535–550

[4] Leistein R.: *Flowshop sequencing with limited buffer storage*. International Journal of Production Research, 28, 1990, 2085–2100

[5] Nowicki E.: *The permutation flow shop with buffers: A tabu search approach.* European Journal of Operational Research, 116, 1999, 205–219

[6] McCormick M.L., Pinedo M.L., Shenker S., Wolf B.: *Sequencing in an assembly line with blocking to minimize cycle time*. Operations Research, 37, 1989, 925–935

[7] Burkowski F.J.: *Proximity and prioryty: applying a gene expresion algorithm to the Traveling Salesperson Problem*. Parralel Computing, 30, 2004, 803–816

[8] Michalewicz Z.: *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin Heidelberg, Spinger-Verlag 1996

[9] Goldberg D.E., Linge R.: *Alleles, Loci and TSP*. Proccedings of the First Internatonal Conference on Genetic Algorithms. Hillsdale, NJ, Lawrence Erlbaum Associates, 1885, 154–159

[10] Allahverdi A., Aldowaisan T.: *No-wait flowshops with bicriteria of makespan and maximum lateness*. European Journal of Operational Research, 152, 2004, 132–142

[11] Taillard E.: *Benchmarks for basic scheduling problems.* European Journal of Operational Research, 64, 1993, 278–285