# NOWE WARIANTY OPERATORÓW GENETYCZNYCH DLA PROBLEMÓW Z KRYTERIUM SUMACYJNYM

**Mariusz MAKUCHOWSKI**

**Streszczenie:** W pracy analizuje się własności sumacyjnego kryterium w permutacyjnym problemie przepływowym. Uzasadnia się trudności optymalizacji badanego problemu, w szczególności tłumaczy się trudności związane z opuszczaniem minimów lokalnych. Przedstawia się pewne propozycje usprawniające proces przeszukiwania przestrzeni rozwiązań w szczególności proponuje się modyfikacje istniejących operatorów genetycznych. Przeprowadza się badania porównawcze algorytmów genetycznych bazujący na klasycznych i proponowanych operatorach. Przeprowadzone badania numeryczne wykonane są na dobrze znanych w literaturze przykładach testowych. Pracę kończy analiza i interpretacja otrzymanych rezultatów.

**Słowa kluczowe:** problem przepływowy, kryterium sumacyjne, operatory genetyczne.

## 1. Wstęp

Problem przepływowy a właściwie jego szczególny przypadek „permutacyjny" jest najprostszym problemem wielomaszynowym w teorii szeregowania zadań. Z praktycznego punktu widzenia jest on niezmiernie ważny gdyż modeluje on klasę procesów produkcyjnych najczęściej występującą w przemyśle. Rozważany problem modeluje proces, w którym zakłada się jednakową marszrutę obrabianych elementów oraz jednakową kolejność obróbki elementów przy każdym stanowisku. Sytuacja taka występuje na wszystkich taśmach produkcyjnych, gdzie obrabiane elementy układane są na taśmie transportującej je do kolejnych stanowisk obróbczych. Ponadto nawet podczas rozwiązywania przypadku ogólniejszego w którym nie jest wymagana jednakowa kolejność wykonywania zadań na poszczególnych maszynach często się taką przyjmuje [1].

Większość prac poświęconych temu zagadnieniu skupia się wokół kryterium minimalizacji momentu zakończenia wszystkich zadań. Fakt ten spowodowany jest tym, iż przy wspomnianym kryterium optymalizacji występuje szereg własności teoretycznych umożliwiających tworzenie na ich bazie bardzo wyspecjalizowanych, efektywnych algorytmów. W przypadku jednak gdy kryterium optymalizacji przyjmuje charakter sumacyjny (minimalizuje się sumę momentów zakończenia zadań) brak jest korzystnych własności problemu. Ponadto przestrzeń rozwiązań staje się wyjątkowo mało podatna na wszelkiego rodzaju algorytmy popraw, ponieważ istnieje bardzo duża liczba optimów lokalnych tworzących skupiska, porozrzucane w całej przestrzeni rozwiązań. Najlepszy algorytm dedykowany temu zagadnieniu jest hybrydą łączącą w sobie elementy algorytmu genetycznego, symulowanego wyżarzania oraz poszukiwania z zabronieniami [2].

Celem pracy jest zaproponowanie oraz przebadanie nowych operatorów genetycznych, (operatory mutacji oraz operatory krzyżowania) szczególnie polecane dla problemów z kryterium sumacyjnym.

## 2. Rozważany problem

Rozważany permutacyjny problem przepływowy z kryterium będącym sumą terminów zakończenia wykonywania zadań, w trójpolowej notacji Graham'a [3], oznaczany jest przez $F^*||C_{sum}$. Jego matematyczny model jest bardzo dobrze znany w literaturze, jednakże ze względu na jego bazowe znaczenie przedstawiony zostanie poniżej.

### 2.1. Model matematyczny problemu

Problem przepływowy można zdefiniować następująco. Dany jest zbiór zadań $J=\{1,2,\ldots,n\}$ oraz zbiór maszyn $M=\{1,2,\ldots,m\}$. Zadanie j, $j\in J$ ma być wykonywane kolejno na maszynach $1,2,\ldots,m$. Czynność polegającą na wykonaniu zadania $j$ na maszynie $l$ nazywamy operacją i notujemy jako parę $(l,j)$. Zadanie $j$ na maszynie $l$ tzn. operacja $(l,j)$ jest realizowana bez przerwy w czasie $p_{l,j}>0$. Przyjmuje się, że

- maszyna $l\in M$ może wykonywać co najwyżej jedną operację w danej chwili,
- nie można jednocześnie wykonywać więcej niż jednej operacji danego zadania,
- każda maszyna wykonuje zadania w tej samej kolejności.

Uszeregowanie dopuszczalne definiowane jest przez momenty rozpoczęcia wykonywania $S(l,j)$ operacji $(l,j)$, $l\in M$, $j\in J$ takie, że spełnione są wszystkie powyższe ograniczenia. Problem polega na znalezieniu takiego uszeregowania dopuszczalnego minimalizującego sumę momentów wykonania wszystkich zadań $\sum_{j\in J} C(m,j)$ gdzie

$$C(l,j)=S(l,j)+p_{l,j}, l\in M, j\in J \ . \qquad (1)$$

Rozwiązania w których nie można zmniejszyć czasu rozpoczęcia żadnego z zadań, tak aby uszeregowanie pozostało nadal rozwiązaniem dopuszczalnym nazywamy rozwiązaniami aktywnymi. (Harmonogram rozwiązań aktywnych dosunięty jest maksymalnie do lewej strony na osi czasu). Przy rozważanym kryterium optymalizacyjnym, rozwiązania optymalne są rozwiązaniami aktywnymi. Dlatego też w dalszej części pracy ograniczono się tylko do analizy rozwiązań aktywnych. Rozwiązanie aktywne można jednoznacznie określić przy pomocy permutacji $\pi=(\pi(1),\pi(2),\ldots,\pi(n))$ wykonywania operacji na maszynach. Zbiór wszystkich możliwych takich permutacji oznaczmy przez $\Pi$. Na podstawie permutacji wykonywania zadań $\pi\in\Pi$ momenty rozpoczęcia poszczególnych operacji możemy wyznaczyć rekurencyjnie ze wzoru:

$$S(l,\pi(j))=\max\{C(l-1,\pi(j)),C(l,\pi(j-1)\},l\in M, j\in J \qquad (2)$$

gdzie $\pi(0)=0$, $C(0,j)=0$, $j\in J$ oraz $C(l,0)=0$, $l\in M$. Niech $C_{sum}(\pi)$ oznacza sumę momentów zakończeń wszystkich zadań w uszeregowaniu aktywnym utworzonym dla permutacji $\pi$,

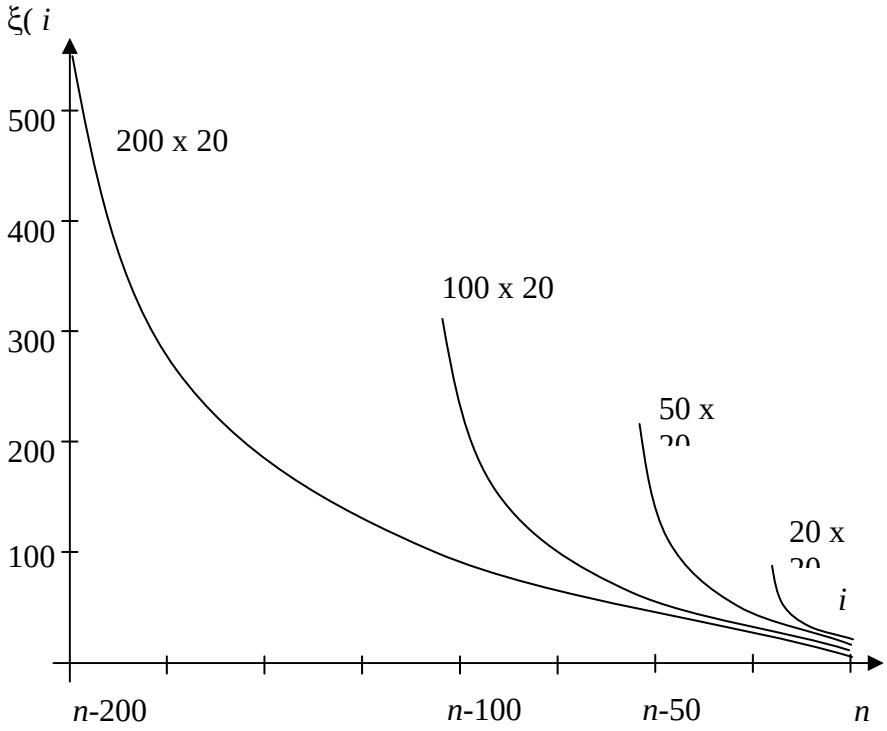
$$C_{sum}(\pi)=\sum_{j\in J} C(m,j) \ . \qquad (3)$$

Ostatecznie rozważany problem sprowadza się do znalezienia $\pi^*\in\Pi$ takiej, że

$$C_{sum}(\pi^*) = \min_{\pi \in \Pi} \{C_{sum}(\pi)\}. \quad (4)$$

### 2.2. Własności numeryczne

W rozdziale tym przedstawione zostaną pewne własności rozważanego problemu bezpośrednio wpływające na jego praktyczną trudność. Prezentowane własności będą wykazane w wyniku przeprowadzenia eksperymentów numerycznych. Posłużą one w dalszej części pracy do stworzenia nowych efektywnych operatorów genetycznych dedykowanych algorytmom genetycznym rozwiązujących problemy z kryterium sumacyjnym.




Rys. 1. Wrazliwość ξ(*i*) pozycji *i* dla instancji o różnej liczbie zadań

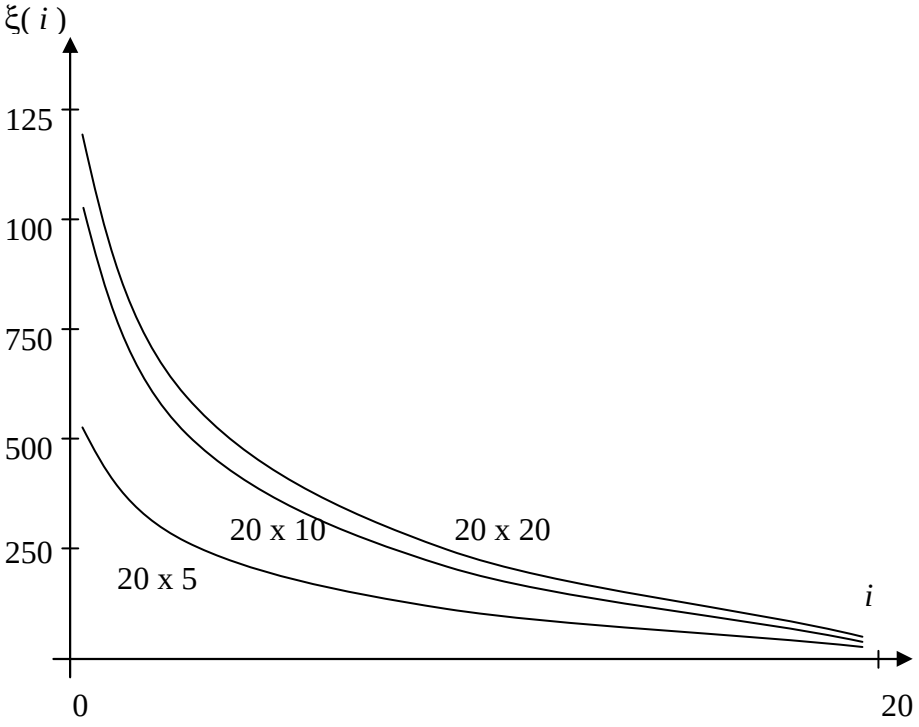
Niech symbol $\pi^i$, $1 \le i < n$ oznacza permutację powstałą z permutacji π poprzez zamianę miejscami elementów *i* i *i*+1;

$$\pi^i = (\pi(1),..., \pi(i-1), \pi(i+1), \pi(i), \pi(i+2),..., \pi(n)). \quad (5)$$

Dodatkowo niech ξ(*i*) oznacza wrażliwość pozycji, rozumiana jako średnia wartość bezwzględnej różnicy wartości funkcji celu losowej permutacji π i $\pi^i$,

$$\xi(i) = AV(|\, C_{sum}(\pi) - C_{sum}(\pi^i)\,|)\ .(6)$$

Rysunek 1 przedstawia wyniki numeryczne wartość wrażliwości pozycji dla czterech przykładów literaturowych różniących się między sobą ilością zadań *n*. Każdy z przykładów opisany jest poprzez podanie rozmiaru *n*×*m*. Ponadto w celu lepszej wizualizacji wszystkie cztery charakterystyki przesunięte są względem siebie w taki sposób by punkt na osi *i* oznaczony przez *n* był rzeczywiście ilością zadań analizowanej instancji. Kolejny rysunek 2 przedstawia wyniki numeryczne wartości wrażliwości pozycji dla trzech przykładów literaturowych przykładów różniących się *m* ilością maszyn. Opis poszczególnych charakterystyk jest dokładnie taki sam jak w przypadku rysunku 1. Dodatkowo tabela 1 zawiera dokładne wartości wrażliwości skrajnych pozycji, ξ(1) i ξ(*n*-1), badanych instancji.




Rys.2. Wrażliwość ξ(*i*) pozycji *i* dla instancji różnej liczbie maszyn

Wynikiem przeprowadzonych badań jest stwierdzenie, iż wrażliwość pozycji permutacji reprezentującej rozwiązanie problemu przepływowego z sumacyjnym kryterium optymalizacji jest silnie zależna od wartości pozycji. W miarę wzrostu rozmiaru instancji, zarówno wzrostu liczby zadań rysunek 1 jak i wzrostu liczby maszyn rysunek 2, różnica wrażliwości pomiędzy pierwszą i ostatnią pozycją w permutacji rośnie. W najmniejszej testowanej instancji 20×05 wrażliwość pierwszej pozycji permutacji π jest ponad dziewięć razy większa niż wrażliwość pozycji ostatniej,ξ(1)/ξ(n-1)= 9,19. W przypadku

największych instancji 500×20 odpowiedni stosunek wynosi już ponad 160, $\xi(1)/\xi(n-1)=160.09$.

Z powyższych własności wynika, iż algorytmy popraw w szczególności algorytmy genetyczne próbujące optymalizować poszczególne części rozwiązania (permutacji) nie są wstanie opuścić minima lokalne związane z początkową i środkową częścią permutacji. Dzieje się tak dlatego, iż w celu opuszczenia minima lokalnego, związanego początkową częścią permutacji, należało by akceptować rozwiązania dużo słabsze niż całe zbiory słabych rozwiązań powstałe w wyniku zmiany końcowej części permutacji.

Tab.1. Wartości wrażliwości skrajnych pozycji testowanych instancji

| przykład | $\xi(1)$ | $\xi(n-1)$ |
|---|---|---|
| 500×20 | 10566 | 66 |
| 200×20 | 5363 | 62 |
| 100×20 | 3143 | 64 |
| 50×20 | 2204 | 71 |
| 20×20 | 1207 | 83 |
| 20×10 | 1022 | 77 |
| 20×05 | 524 | 57 |

Powyższa bardzo niekorzystna własność oraz brak pozytywnych cech problemu występujących przy innych kryteriach optymalizacji, takich jak własności blokowe [4,5] czy możliwość stosowania bardzo wyspecjalizowanych akceleratorów [6] powodują iż rozważany problem jest bardzo trudny do rozwiązania.

## 3. Algorytm genetyczny

Na wstępie należy, podkreślić iż wszystkie algorytmy genetyczne, niezależnie od problemu dla którego zostały adaptowane bazują na wspólnej idei. U podstaw każdego algorytmu genetycznego [7] leży Darwinowska teoria ewolucji występująca w świecie przyrody. Teoria ta mówi, iż w wyniku selekcji naturalnej oraz mechanizmów dziedziczenia dana populacja ma tendencje do generowania z pokolenia na pokolenie coraz lepiej przystosowanych osobników. Algorytmy genetyczne symulują więc zarówno środowisko jak i życie wirtualnych osobników. Każdy z osobników identyfikowany jest z jednym rozwiązaniem, a jakość przystosowania oceniana się na podstawie wartości optymalizowanej funkcji. Po zakończeniu obliczeń, algorytm dostarcza rozwiązanie odpowiadające najlepszemu osobnikowi który pojawił się podczas całej symulacji. Właściwie dobrane parametry algorytmu genetycznego w szczególności odpowiednio dobrany mechanizm dziedziczenia wraz z selekcją promującą osobniki o żądanej cesze gwarantują iż w symulowanym świecie pojawia się ewolucja. Zmieniając definicję przystosowania czyli sposobu oceny danego osobnika możemy dokonać wyboru kierunku ewolucji. W praktyce wartość przystosowania danego osobnika przyjmuje wartość optymalizowanej funkcji celu odpowiadającego mu rozwiązania. Jedna iteracja algorytmu genetycznego jest zazwyczaj symulacją pojedynczego pokolenia, w którym osobniki najsłabiej przystosowane giną zazwyczaj bezpotomnie, natomiast najlepiej przystosowane stają się rodzicami nowego pokolenia. Nowo powstałe osobniki dziedziczą geny (pewne atrybuty rozwiązań) swoich rodziców. Dzięki temu powstałe rozwiązania są kombinacją

najlepszych rozwiązań pokolenia wcześniejszego. Ponadto każdy z algorytmów genetycznych, wyposażony jest w mechanizm mutacji wprowadzający niewielkie zaburzenia wśród powstających osobników. Ma to na celu uniknięcie zwyrodnienia pokolenia w którym wszystkie nowo generowane osobniki są do siebie bardzo podobne. Ponadto dzięki mutacji algorytm bada coraz to nowe obszary przestrzeni rozwiązań co sprzyja opuszczaniu minimów lokalnych oraz umożliwia wygenerowanie osobnika posiadającego pewną cechę nie występującą całej populacji. Sprzyja to zarówno osiągnięciu przez algorytm minimum lokalnego, jak i późniejszym jego opuszczeniu.

Z powyższego opisu wynika, iż w każdym algorytmie genetycznym można wyróżnić następujące bazowe elementy [8]:

- *generacja populacji startowej*: generacja osobników pierwszego pokolenia,
- *selekcja*: wybór rodziców z całego pokolenia
- *krzyżowanie*: generowanie nowych osobników na podstawie genów rodziców
- *mutacja*: wprowadzenie niewielkich zmian w genach nowo powstałych osobników
- *kryterium stopu*: określa warunek zatrzymujący pracę algorytmu

Konkretne implementacje algorytmów genetycznych dedykowane poszczególnym problemom kombinatorycznym różnią się między sobą przede wszystkim sposobem kodowania rozwiązania oraz sposobem obliczania funkcji przystosowania (na podstawie optymalizowanego kryterium). Ponadto w danym algorytmie genetycznego należy sprecyzować sposób realizacji wymienionych wyżej pięciu bazowych elementów. W dalszej części pracy przedstawione zostaną nowe sposoby krzyżowania oraz mutacji szczególnie polecane dla problemów z sumacyjną funkcją celu.

### 3.1. Nowe operatory genetyczny

Pod pojęciem operatorów genetycznych kryją się zarówno operatory krzyżowania jak i operatory mutacji. Wyniku działania tych operatorów powstaje zawsze nowy osobnik reprezentujący pewne rozwiązanie danego problemu. Następnie każdy nowo powstały osobnik oceniany jest na podstawie wartości funkcji celu odpowiadającego mu rozwiązania.

W przypadku klasycznych operatorów krzyżowania (przykładowo PMX) nowo powstały osobnik (permutacja) w pierwszej części jest podobny do pierwszego rodzica, natomiast część druga wzorowana jest na drugim z rodziców. Punkt podziału tworzonego osobnika wybierany jest losowo z równomiernym prawdopodobieństwem. Podobnie jest podczas klasycznych operatorów mutacji, które wprowadzają niewielkie zaburzenia w genotypie osobnika. Zaburzenie te jakiejkolwiek postaci (zależnej od stosowanego operatora mutacji) zmienia pewną część permutacji. Zmieniana część permutacji wybierana jest także z równomiernym prawdopodobieństwem. Tak tworzone osobniki poddawane są dalej selekcji która kieruje się tylko wartością funkcji przystosowania, nie uwzględniająca położenie punktu krzyżowania podczas tworzenia potomka ani też miejsca wprowadzonej mutacji. Jednakże w rozważanym problemie z kryterium sumacyjnym powyższe postępowanie wydaje się nieodpowiednie. Ze względu na duże różnice w wrażliwości pozycji (patrz właściwości problemu), aby opuścić minimum lokalne związane z początkową częścią permutacji należy przejść przez rozwiązania znacznie słabsze niż tysiące rozwiązań powstających wyniku zmiany końcowych elementów permutacji. Ponieważ selekcja nie uwzględnia miejsca modyfikacji rozwiązania, a kieruje się tylko wartością funkcji celu, prawdopodobieństwo opuszczenia minimum lokalnego związanego z początkową częścią permutacji jest niezwykle małe, bliskie zeru. W celu umożliwienia

opuszczania minimów lokalnych z wiązanych z początkową częścią permutacji należy dodatkowo promować modyfikacje rozwiązań na niższych pozycjach w permutacji. Mechanizm taki można zaszyć w metodę selekcji, jednakże w bieżącej pracy proponuję się pewną modyfikację stosowanych operatorów genetycznych.

Ogólnie rzecz ujmując, proponowana modyfikacja operatorów genetycznych dla problemu z kryterium sumacyjnym polega na generowaniu dużej liczby osobników, z modyfikowaną częścią rozwiązania o dużej wrażliwości oraz małą liczbą osobników z modyfikowaną częścią rozwiązania o wrażliwości niewielkiej. Można tego dokonać poprzez zmianę rozkładu prawdopodobieństwa wyboru punktów krzyżowania oraz zmianę rozkładu prawdopodobieństwa mutowanych fragmentów rozwiązania. Oczywiste jest to, że pozycje o większej wrażliwości powinny być modyfikowane znacznie częściej, niż pozycje o wrażliwości niskiej. Wtedy bowiem istnieje szansa iż algorytm opuści minimum lokalne z wiązane z częścią permutacji o wysokiej wrażliwości.

### 3.2. Badania numeryczne

Badania numeryczne przeprowadzone zostały na dobrze znanych przykładach literaturowych [9]. Dla każdego z testowanych przykładów podana jest wartość referencyjna zaczerpnięta z pracy [10] oraz wartość rozwiązania uzyskanego algorytmem konstrukcyjnym *NEH* [11]. Następnie dla każdego z przykładów uruchomiony został klasyczny algorytm genetyczny oznaczany *GA* z typowymi operatorami genetycznymi oraz algorytm genetyczny *GA** z operatorami promującymi modyfikacje rozwiązań w miejscach dużej wrażliwości. Konkretnie w algorytmie *GA* zastosowano wielopunktowy operator krzyżowania bazujący na idei operatora PMX oraz mutację bazującą na operatorze INSERT. Operator INSERT przekłada losowo wybrany element permutacji w nowe położenie. Algorytmie *GA** różni się od algorytmu *GA* tylko sposobem losowania punktów krzyżowania. Punkty te klasycznie losowane są z prawdopodobieństwem o równomiernym rozkładzie, natomiast po modyfikacji rozkład ten zostaje zmieniony na rozkłady wykładniczy (z dobranymi na stałe parametrami).

Ponadto dla każdego przykładu na podstawie wartości funkcji celu, wygenerowanych rozwiązań, wyznaczono błąd względem rozwiązań referencyjnych,

$$\rho^A = 100\% \cdot \frac{C_{\max}{}^{A} - C_{\max}{}^{ref}}{C_{\max}{}^{ref}}, A \in \{NEH, GA, GA^*\}. \qquad (7)$$

Instancje charakteryzujące się jednakowym rozmiarem, tj. jednakową liczbą zadań *n* oraz jednakową liczbą maszyn *m*, tworzą grupę oznaczaną przez *n*×*m*. Dla każdej testowanej grupy podana jest wartość średnia odpowiednich błędów względnych. Uzyskane wartości funkcji celu $C_{sum}$ poszczególnych rozwiązań oraz błędy względne ρ dla pierwszych 30 instancji Taillard'a podane są w tabeli 2 dla kolejnych 20 instancji zawarte są w tabeli 3.

Wszystkie prezentowane testy przeprowadzone został na komputerze klasy PC wyposażonym w procesor Athlon 2000+ (1667MHz) w środowisku wielozadaniowym Windows XP. Opisane algorytmy zostały zaprogramowanie w języku C++ i skompilowane przez Dev C++ w wersji 4.9.9.1. Całkowity czas obliczeń wszystkich testowanych przykładów (50 instancji) dla algorytmu *GA* wynosi 6 s natomiast dla algorytmu *GA** wynosił 10 s.

Tab.2. Wartości $C_{sum}$ poszczególnych algorytmów oraz błąd względny ρ[%] dla instancji ta001-ta030

| Przykład | $C_{sum}^{ref}$ | $C_{sum}^{NEH}$ | $C_{sum}^{GA}$ | $C_{sum}^{GA*}$ | $\rho^{NEH}$ | $\rho^{GA}$ | $\rho^{GA*}$ |
|---|---|---|---|---|---|---|---|
| ta001 | 14033 | 14659 | 14536 | 14071 | 4,46 | 3,58 | 1,71 |
| ta002 | 15151 | 16593 | 15549 | 15410 | 9,52 | 2,63 | 0,35 |
| ta003 | 13301 | 15321 | 13860 | 13347 | 15,19 | 4,20 | 1,48 |
| ta004 | 15447 | 16974 | 15824 | 15676 | 9,89 | 2,44 | 0,50 |
| ta005 | 13529 | 14383 | 14030 | 13596 | 6,31 | 3,70 | 0,00 |
| ta006 | 13123 | 15344 | 13533 | 13123 | 16,92 | 3,12 | 1,57 |
| ta007 | 13548 | 15639 | 13863 | 13761 | 15,43 | 2,33 | 1,38 |
| ta008 | 13948 | 15704 | 14503 | 14140 | 12,59 | 3,98 | 2,03 |
| ta009 | 14295 | 16061 | 15130 | 14585 | 12,35 | 5,84 | 1,96 |
| ta010 | 12943 | 14618 | 13353 | 13197 | 12,94 | 3,17 | 1,12 |
| | | | | | 11,56 | 3,50 | 1,71 |
| ta011 | 20911 | 23101 | 21831 | 21128 | 10,47 | 4,40 | 1,04 |
| ta012 | 22440 | 24822 | 23533 | 22740 | 10,61 | 4,87 | 1,34 |
| ta013 | 19833 | 22053 | 20470 | 19985 | 11,19 | 3,21 | 0,77 |
| ta014 | 18710 | 20572 | 19280 | 18932 | 9,95 | 3,05 | 1,19 |
| ta015 | 18641 | 20343 | 19139 | 18838 | 9,13 | 2,67 | 1,06 |
| ta016 | 19245 | 21010 | 19884 | 19706 | 9,17 | 3,32 | 2,40 |
| ta017 | 18363 | 20392 | 19078 | 18419 | 11,05 | 3,89 | 0,30 |
| ta018 | 20241 | 22352 | 20993 | 20326 | 10,43 | 3,72 | 0,42 |
| ta019 | 20330 | 22183 | 21152 | 20564 | 9,11 | 4,04 | 1,15 |
| ta020 | 21320 | 23317 | 22052 | 21380 | 9,37 | 3,43 | 0,28 |
| | | | | | 10,05 | 3,66 | 0,99 |
| ta021 | 33623 | 37157 | 34724 | 33913 | 10,5 | 3,2 | 0,8 |
| ta022 | 31587 | 34052 | 32339 | 31597 | 7,80 | 2,38 | 0,03 |
| ta023 | 33920 | 38251 | 34720 | 34138 | 12,7 | 2,3 | 0,6 |
| ta024 | 31661 | 33315 | 32490 | 31788 | 5,22 | 2,62 | 0,40 |
| ta025 | 34557 | 37190 | 35313 | 34830 | 7,62 | 2,19 | 0,79 |
| ta026 | 32564 | 34487 | 33615 | 32647 | 5,91 | 3,23 | 0,25 |
| ta027 | 32922 | 35479 | 33825 | 33429 | 7,77 | 2,74 | 1,54 |
| ta028 | 32412 | 34881 | 33042 | 32711 | 7,62 | 1,94 | 0,92 |
| ta029 | 33600 | 35939 | 34630 | 33939 | 6,96 | 3,07 | 1,01 |
| ta030 | 32262 | 35107 | 33570 | 32760 | 8,82 | 4,05 | 1,54 |
| | | | | | 8,10 | 2,79 | 0,80 |

Tab.3. Wartości $C_{sum}$ poszczególnych algorytmów oraz błąd względny ρ[%] dla instancji ta031-ta050

| Przykład | $C_{sum}^{ref}$ | $C_{sum}^{NEH}$ | $C_{sum}^{GA}$ | $C_{sum}^{GA*}$ | $\rho^{NEH}$ | $\rho^{GA}$ | $\rho^{GA*}$ |
|---|---|---|---|---|---|---|---|
| ta031 | 64860 | 76071 | 71232 | 66079 | 17,28 | 9,82 | 1,88\ |
| ta032 | 68134 | 78867 | 73886 | 69592 | 15,75 | 8,44 | 2,14 |
| ta033 | 63304 | 72291 | 70816 | 66190 | 14,20 | 11,87 | 4,56 |
| ta034 | 68259 | 80328 | 74478 | 70786 | 17,68 | 9,11 | 3,70 |
| ta035 | 69491 | 76897 | 75355 | 70907 | 10,66 | 8,44 | 2,04 |
| ta036 | 67006 | 78506 | 73055 | 68484 | 17,16 | 9,03 | 2,21 |
| ta037 | 66311 | 79953 | 70789 | 68090 | 20,57 | 6,75 | 2,68 |
| ta038 | 64412 | 77185 | 70774 | 66217 | 19,83 | 9,88 | 2,80 |
| ta039 | 63156 | 76247 | 67521 | 65465 | 20,73 | 6,91 | 3,66 |
| ta030 | 68994 | 80669 | 76378 | 71857 | 16,92 | 10,70 | 4,15 |
| | | | | | 17,08 | 9,10 | 2,98 |
| ta041 | 87430 | 101299 | 96053 | 91577 | 15,86 | 9,86 | 4,74 |
| ta042 | 83157 | 93469 | 92398 | 86841 | 12,40 | 11,11 | 4,43 |
| ta043 | 79996 | 94149 | 92036 | 84370 | 17,69 | 15,05 | 5,47 |
| ta044 | 86725 | 99324 | 98463 | 90277 | 14,53 | 13,53 | 4,10 |
| ta045 | 86448 | 95967 | 95192 | 91232 | 11,01 | 10,11 | 5,53 |
| ta046 | 86651 | 95791 | 94871 | 90717 | 10,55 | 9,49 | 4,69 |
| ta047 | 89042 | 98201 | 99227 | 92024 | 10,29 | 11,44 | 3,35 |
| ta048 | 86924 | 100119 | 96250 | 90790 | 15,18 | 10,73 | 4,4 |
| ta049 | 85674 | 94281 | 93955 | 88798 | 10,05 | 9,67 | 3,65 |
| ta050 | 88215 | 104036 | 96210 | 91041 | 17,93 | 9,06 | 3,20 |
| | | | | | 13,55 | 11,01 | 4,36 |

## 4. Podsumowanie

Z przeprowadzonych badań wynika, iż dobierając sposób losowania punktów krzyżowania, można w łatwy sposób zwiększyć efektywność operatorów genetycznych. W rozważanym problemie zmieniając rozkład losowania tych punktów, średni błąd algorytmu względem rozwiązań referencyjnych zmniejszył się z 6.01% do 2.05%, przy stosunkowo niewielkim wzroście czasu działania algorytmu (z 6s do 10s). Otrzymana poprawa jest większa nawet niż w przypadku stokrotnego zwiększenia liczby symulowanych pokoleń (w tym przypadku średni błąd zmalał do 4.08% a czas obliczeń zwiększył się z 6s do 600s).

Prezentowane podejście można stosować we wszystkich algorytmach genetycznych w których poszczególne części rozwiązania mają różny średni wpływ na zmianę wartości funkcji celu. Zjawisko te zachodzi między innymi w problemach szeregowania zadań, z sumacyjnym kryterium optymalizacji.

**Literatura**


1. C. Smutnicki, Algorytmy szeregowania, Akademicka opicyna wydawnicza Exit, Warszawa 2002.
2. T. Yamada, C.R. Reeves, Solving the Csum Prmutation Flowshop Scheduling Problem by Genetic Local Search, International Conference on Evolutionary Computation, 230-234, 1998.
3. R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: survey, Annals of Discrete Mathematics, 5:287-326, 1979.
4. J. Grabowski, J. Pempera, New block properties for the permutation flowshop problem with aplication in TS, Journal of Operational Research No. 52, 210-220, 2001.
5. J. Grabowski, E. Nowicki, C. Smutnicki, Metoda blokowa w zagadnieniach szeregowania
6. E. Nowicki, C. Smutnicki, A fast tabu search algorithm for the permutation flow shop problem, European Journal of Operational Research No. 91, 160-175, 1996.
7. J.H. Holland, Genetic Algorithms, Scientific American, 44, 1992.
8. D.E. Goldberg, Algorytmy genetyczne i ich zastosowania, WNT Warszawa, 1995
9. E. Taillard, Benchmarks for basic scheduling problems, European Journal of Operational Research, 278-285, 1993.
10. T. Yamada, C.R. Reeves, Solving the Csum Prmutation Flowshop Scheduling Problem by Genetic Local Search, International Conference on Evolutionary Computation, 230-234, 1998.
11. M. Nawaz, Jr.E. Enscore, I. Ham, A heuristic algorithm for the m-machine, n-job flow-



Dr inż. Mariusz MAKUCHOWSKI
Politechnika Wrocławska,
Instytut Informatyki Automatyki i Robotyki
50-372 Wrocław, ul Janiszewskiego 11/17
tel.: (0-71) 320-28-34,
e-mail: mariusz.makuchowski@pwr.wroc.pl