AUTOMATYKA • 2007 • Tom 11 • Zeszyt 1–2

Adam Tyński*

Zastosowanie nowego operatora krzyżowania w rozwiązywaniu problemu gniazdowego z transportem

1. Wprowadzenie

Oczywisty wymóg zwiększania wydajności elastycznych systemów produkcyjnych przekłada się między innymi na:

(i) zwiększenie precyzji modelowania ich pracy,
(ii) zwiększenie efektywności algorytmów szeregowania wykonywanych w nich zadań.

Zwiększenie precyzji modelowania można osiągnąć poprzez uwzględnienie dodatkowych ograniczeń technologicznych, takich jak np. skończona liczba i pojemność buforów, niezerowe czasy przezbrojeń czy (rozważane w tej pracy) czasy transportu pomiędzy maszynami produkcyjnymi. Ze względu na NP-trudność zdecydowanej większości problemów szeregowania zadań, w ostatnich latach dynamiczny rozwój przeżywają algorytmy heurystyczne, w szczególności algorytmy lokalnych poszukiwań. Efektywność tych algorytmów można zwiększyć np. poprzez zaopatrzenie ich w odpowiednie mechanizmy eksploracji szczególnie obiecujących obszarów przestrzeni rozwiązań. Zachowanie odpowiedniego balansu pomiędzy dywersyfikacją oraz intensyfikacją obliczeń mogą zapewnić hybrydowe algorytmy genetyczne wykorzystujące np. nowoczesne operatory krzyżowania i algorytmy poszukiwań z zabronieniami.

W pracy został przedstawiony nowy quasi-operator krzyżowania, nazwany operatorem *MX*. Z operatora wyeliminowano wiele wad innych operatorów tego typu opisanych w literaturze przy jednoczesnym zachowaniu ich najważniejszych zalet. Przykładem może tu być zachowanie idei „ścieżki łączącej" przy jednoczesnym uproszczeniu parametryzacji operatora (poprzez ograniczenie do minimum liczby wymaganych parametrów). Operator został osadzony w hybrydowym algorytmie genetycznym, wykorzystującym w trakcie swojej pracy m. in. algorytm poszukiwań z zabronieniami. Algorytm genetyczny został użyty do heurystycznego rozwiązania uogólnienia problemu gniazdowego, w którym każde zadanie należy dodatkowo przetransportować pomiędzy jednomaszynowymi gniazdami produkcyjnymi. Każdego transportu należy dokonać przy użyciu jednego z dostępnych w systemie wózków AGV, których liczba jest ograniczona. Zakłada się, że przydział

* Instytut Informatyki, Automatyki i Robotyki, Politechnika Wrocławska

transportów do poszczególnych wózków nie jest znany i stanowi dodatkową zmienną decyzyjną. Zakłada się też, że wózki pracują w trybie „zabierz i zostaw", co oznacza, że po zostawieniu transportowanego detalu w danym gnieździe produkcyjnym wózek może wykonać tzw. przejazd pusty (przejazd bez załadunku) do innego gniazda i tam rozpoczyna kolejny transport. Problem polega na odnalezieniu takiej kolejności wykonania poszczególnych operacji i transportów, by kryterium optymalizacji – moment zakończenia wykonywania wszystkich zadań – przyjęło wartość minimalną.

Przedstawiony w tej pracy algorytm został poddany badaniom numerycznym przy użyciu odpowiednio dobranych instancji testowych. W wyniku przeprowadzonych analiz okazuje się, że algorytm ten dostarcza wysokiej jakości rozwiązań badanego problemu.

2. Model matematyczny

Model matematyczny problemu był już omawiany, np. w pracy [1], dlatego poniżej przedstawia się tylko jego najważniejsze elementy. Dany jest zbiór m^p maszyn produkcyjnych $M^p = \{1, 2, ..., m^p\}$ i zbiór m^t maszyn transportowych (utożsamianych z wózkami AGV) $M^t = \{m^p+1, m^p+2, ..., m^p+m^t\}$; $M = M^p \cup M^t$, $m = |M| = m^p + m^t$. Dany jest zbiór r zadań $J = \{1, 2, ..., r\}$. Każde zadanie $k \in J$ składa się z sekwencji n_k operacji produkcyjnych, indeksowanych przez $j_k+1, j_k+2, ..., j_k+n_k$, które powinny być wykonane w tej kolejności; $j_k = \sum_{i=1}^{k-1} n_i$, $j_1 = 0$. Przez $O^p = \{1, 2, ..., n^p\}$ będzie oznaczany zbiór wszystkich operacji produkcyjnych, gdzie $n^p = \sum_{k\in J} n_k$. Każdą operację produkcyjną $a \in O^p$ należy wykonać na maszynie $m(a) \in \mu(a) \subseteq M^p$ w czasie $p^a > 0$, przy czym przyjmuje się, że $|\mu(a)| = 1$. Zakłada się, że każde dwie kolejne operacje produkcyjne tego samego zadania nie mogą być wykonywane na tej samej maszynie produkcyjnej.

Transport zadania $k \in J$ i przejazd pusty maszyny transportowej pomiędzy maszynami $x, y \in M^p$, $x \neq y$, wymaga odpowiednio $t_k(x, y) > 0$ i $e(x, y) > 0$ czasu; $t_k(x, x) = e(x, x) = 0$. Zakłada się, że czasy transportu i przejazdów pustych spełniają silny warunek trójkąta oraz zależność $t_k(x, y) \geq e(x, y)$, $x, y \in M^p$. Pomiędzy każdą parą sąsiednich operacji produkcyjnych $j_k + i$ oraz $j_k + i + 1$, $1 \leq i < n_k$, zadania $k \in J$ jest wykonywana operacja transportowa, oznaczana symbolem $a = [j_k + i]$. Operacja a powinna być wykonana przez dowolną maszynę transportową $m(a) \in \mu(a) \subseteq M^t$ w czasie $p_a = t_k(x, y)$, gdzie $x = m(j_k + i)$ oraz $y = m(j_k + i + 1)$. Zakłada się, że czas transportu p_a zadania a nie zależy od maszyny $m(a) \in \mu(a)$. Zbiór wszystkich operacji transportowych będzie oznaczany przez $O^t = \bigcup_{k\in J} \bigcup_{i=1}^{n_k-1} \{[j_k + i]\}$. Całkowita liczba operacji transportowych wynosi $n^t = |O^t| = \sum_{k\in J} (n_k - 1) = n^p - r$. W celu wprowadzenia kompletnej notacji definiuje się zbiór wszystkich operacji $O = O^p \cup O^t$ oraz liczbę wszystkich operacji $n = |O| = n^p + n^t$. Po wykonaniu operacji $a = [j_g + i]$, $1 \leq i < n_g$, $g \in J$, i przed wykonaniem operacji $b = [j_k + h]$, $1 \leq h < n_k$, $k \in J$, maszyna transportowa wykonuje przejazd pusty pomiędzy maszynami $x = m(j_g + i + 1)$, $y = m(j_k + h)$, który trwa $e(x, y)$ jednostek czasu. Przejazd pusty wygodnie jest utożsamić z przezbrojeniem $s(a, b) = e(x, y)$ maszyny transportowej, wykonywanym pomiędzy operacjami a, b. W ten sposób zdefiniowane czasy przezbrojeń spełniają słaby warunek trójkąta.

Uszeregowaniem będzie nazywany zbiór par $S = \{(S_1, m(1)), (S_2, m(2)), ..., (S_n, m(n))\}$, gdzie $m(i) \in \mu(i)$ jest maszyną wybraną do wykonania operacji $i \in O$, zaś S_i jest czasem rozpoczęcia jej wykonywania. Uszeregowanie jest dopuszczalne, gdy spełnia ograniczenia typowe dla klasycznego problemu gniazdowego. Problem polega na odnalezieniu takiego uszeregowania dopuszczalnego, by moment zakończenia wykonywania procesu technologicznego, równy $\max_{i \in O} S_i + p_i$, przyjął wartość minimalną. Problem jest silnie NP-trudny.

3. Reprezentacja rozwiązań i miary odległości

W algorytmach genetycznych przestrzeń rozwiązań (zbiór wszystkich uszeregowań) jest odwzorowywana w zbiór ciągów kodowych skończonej długości (zwanych osobnikami bądź chromosomami). Dla problemów gniazdowych znane jest przynajmniej dziewięć reprezentacji rozwiązań, spośród których jedną z najbardziej popularnych wydaje się reprezentacja bazująca na listach preferencyjnych. W tym podejściu uszeregowanie jest odwzorowywane w zestaw permutacji podziału rozłącznego zbioru operacji, opisany poniżej. Niech $\theta = \{O_1, O_2, ..., O_m\}$ oznacza podział rozłączny zbioru O taki, że $\forall_{k \in O} \exists!_{l \in \mu(k)}\ k \in O_l$. Niech Θ określa zbiór wszystkich podziałów rozłącznych zbioru O spełniających powyższy warunek. Nie trudno zauważyć, że dla każdego podziału $\theta \in \Theta$ zbiory $O_1, ..., O_{m^p}$ są takie same, natomiast zbiory $O_{m^p+1}, ..., O_m$ stanowią dowolny podział rozłączny zbioru O^t. Niech $\pi = (\pi_1, ..., \pi_m)$ będzie zestawem permutacji podziału rozłącznego zbioru O, gdzie $\pi_l = (\pi_l(1), ..., \pi_l(o_l)), o_l = |O_l|$, jest permutacją zbioru O_l, określającą preferowaną kolejność (listę preferencyjną) wykonania operacji z tego zbioru na maszynie l. Niech Π^0 będzie zbiorem wszystkich zestawów permutacji. Dla uproszczenia, w dalszej części pracy zestaw permutacji $\pi \in \Pi^0$ będzie nazywany permutacją bądź po prostu rozwiązaniem problemu. Permutacja π reprezentująca uszeregowanie dopuszczalne będzie nazywana permutacją dopuszczalną (rozwiązaniem dopuszczalnym), zaś zbiór wszystkich permutacji dopuszczalnych będzie oznaczany symbolem $\Pi \subset \Pi^0$. Symbolem $\theta(\pi) \in \Theta$ będzie oznaczany podział rozłączny, którego permutacją jest permutacja $\pi \in \Pi^0$.

W dalszych rozważaniach istotne będzie pojęcie odległości pomiędzy permutacjami. Przedstawiona poniżej miara odległości, omawiana już wcześniej, np. w pracy [2], bazuje na reprezentacji permutacji w n-wymiarowej przestrzeni Euklidesowej. Mianowicie, dla dowolnej permutacji $\pi \in \Pi^0$ można określić punkt $A = (a_1, a_2, ..., a_n) \in R^n$ taki, że $a_{\varphi(i)} = \pi^{-1}(i)$, $i \in O$, gdzie $\varphi: O \to \{1, 2, ..., n\}$ jest dowolną funkcją odwzorowującą zbiór operacji w podzbiór zbioru liczb naturalnych. Symbol π^{-1} oznacza tutaj permutację odwrotną do permutacji π, tj. spełnia równanie $\pi_{m(i)}(\pi^{-1}(i)) = i$ dla każdej operacji $i \in O$. Możliwe jest zatem zdefiniowanie miary odległości

$$e(\pi, \sigma) = \| AB \| = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \tag{1}$$

pomiędzy permutacjami $\pi, \sigma \in \Pi^0$ równej odległości euklidesowej pomiędzy punktami $A = (a_1, a_2, ..., a_n)$, $B = (b_1, b_2, ..., b_n)$, reprezentującymi owe permutacje w przestrzeni R^n.

Wiele własności przedstawionej wyżej miary geometrycznej pozostaje nieznane. Należy tu jednak podkreślić, że w przeciwieństwie do wielu innych powszechnie używanych miar odległości miara ta jest poprawnie określona dla dowolnej pary permutacji $(\pi, \sigma) \in \Pi^0$, tzn. również w sytuacji, w której $\theta(\pi) \neq \theta(\sigma)$. Faktem jest też istnienie istotnej dodatniej korelacji pomiędzy miarą geometryczną i często używaną miarą tau Kendalla (oczywiście tylko w przypadku tych par permutacji (π, σ), dla których miara tau Kendala jest określona, tzn. gdy $\theta(\pi) = \theta(\sigma)$). Można zatem się spodziewać, że wiele rezultatów prawdziwych dla miary tau Kendalla jest też prawdziwe dla miary geometrycznej.

4. Operator krzyżowania i algorytm genetyczny

W algorytmie genetycznym podczas krzyżowania z dwóch chromosomów macierzystych formowane są chromosomy potomne wskutek wymiany miejscami ich fragmentów. W przypadku reprezentacji rozwiązań bazującej na listach preferencyjnych, w procesie krzyżowania konieczne jest użycie odpowiednich operatorów. Użycie operatorów krzyżowania gwarantuje tzw. **legalność** chromosomów potomnych, tzn. w wyniku krzyżowania dwóch permutacji zawsze powstaje permutacja. Operatory krzyżowania powinny być też dobrane w ten sposób, by ich użycie pozwoliło na dostateczną eksplorację przestrzeni rozwiązań. W przypadku problemów szeregowania zadań dobrą alternatywą wydaje się stosowanie quasi-operatorów rekombinacji wykorzystujących ideę „ścieżki łączącej" (*path relinking*). W myśl tej idei, operator krzyżowania posiada cechy algorytmu lokalnych poszukiwań poszukującego najlepszych rozwiązań znajdujących się na ścieżce łączącej dwa krzyżowane ze sobą chromosomy macierzyste. Taka strategia poszukiwań ma szczególne znaczenie w przypadku problemów, dla których wykazano obecność tzw. „wielkiej doliny" w krajobrazie ich przestrzeni rozwiązań (patrz. np. [3]). Ścieżka łącząca może być zdefiniowana na wiele sposobów, jak również można stosować wiele strategii jej przemierzania. Jednymi z najbardziej znanych quasi-operatorów są operatory *MSXF* oraz *MSMF* [3]. Ich zaletą jest to, że w trakcie swojej pracy generują tylko permutacje dopuszczalne. Jednakże ich użycie wiąże się z koniecznością doboru szeregu parametrów i elementów składowych, zależnych od specyfiki rozwiązywanego problemu. Konieczność ta znacznie utrudnia użycie operatorów w praktyce, a w przypadku niektórych problemów czyni operatory stosunkowo mało efektywnymi. Dlatego uzasadnione są próby projektowania nowych operatorów o podobnej zasadzie działania, których efektywność nie jest determinowana przez dobór elementów zależnych od specyfiki problemu.

W pracy [2] został przedstawiony quasi-operator krzyżowania *GX*, który pierwotnie był wykorzystywany w rozwiązywaniu klasycznego problemu gniazdowego z kryterium C_{sum}. W operatorze tym, podobnie jak w operatorach *MSXF* i *MSMF*, wykorzystuje się ideę „ścieżki łączącej". Wykorzystuje się też geometryczną miarę odległości pomiędzy permutacjami, daną równaniem (1). Aby zastosować operator *GX* w algorytmie rozwiązywania badanego problemu, należy go jednak poddać pewnej modyfikacji; w klasycznym problemie gniazdowym zakłada się, że przydział wszystkich operacji do maszyn jest z góry ustalony. Oznacza to, że moc zbioru podziałów rozłącznych spełniających warunek $\forall_{k \in O} \exists!_{l \in \mu(k)}\ k \in O_l$ wynosi wtedy $|\Theta| = 1$, co oczywiście nie jest prawdą w przypadku problemu opisywanego w tej pracy.

Rozważmy dwie permutacje $\pi, \sigma \in \Pi^0$ takie, że $\theta(\pi) \neq \theta(\sigma)$. Istotą modyfikacji operatora *GX* jest umożliwienie jego zastosowania do permutacji π oraz σ. Z założenia, w wyniku wykonania jednego z kroków każdej z *maxiterX* iteracji będzie formowany zestaw permutacji δ podziału $\theta(\delta) = \{O_1, O_2, ..., O_m\}, o_l = |O_l|, l \in M$, gdzie *maxiterX* jest parametrem. Należy zatem znaleźć metodę konstrukcji podziału $\theta(\delta)$. Najbardziej naturalnym rozwiązaniem wydaje się być dodanie każdej operacji $i \in O$ z równym prawdopodobieństwem do zbioru O_k albo zbioru O_l, gdzie $k = m^{\pi}(i)$, $l = m^{\sigma}(i)$. Symbole $m^{\pi}(i)$, $m^{\sigma}(i)$ oznaczają tu maszyny wybrane do wykonania operacji i w odpowiedniej permutacji π oraz σ. Operator *MX*, powstały w wyniku wprowadzenia powyższej modyfikacji do operatora *GX*, został przedstawiony poniżej.

Schemat operatora *MX*

Operator rozpoczyna działanie z ustalonymi permutacjami π, σ i parametrem *maxiterX*. Operator zwraca permutację dopuszczalną π^* i wartość funkcji celu C^*.

Krok 1. Określ funkcję φ i znajdź punkty $A, B \in R^n$ odpowiadające permutacjom π oraz σ. Połóż $A' := A, B' := B, \pi^* := \emptyset, C^* := \infty$ oraz $iter := 0$. Połóż $O_i := \emptyset, i = 1, ..., m$.

Krok 2. Dla każdej operacji $k \in O$ wykonaj krok 3.

Krok 3. Wylosuj liczbę r z przedziału (0, 1). Jeżeli $r \leq 0{,}5$, to połóż $m(k) := m^{\pi}(k)$. W przeciwnym przypadku połóż $m(k) := m^{\sigma}(k)$. Połóż $O_{m(k)} := O_{m(k)} \cup k$.

Krok 4. Połóż $W_i := \{1, ..., o_i\}$, $i = 1, ..., m$. Utwórz listę $\gamma = (\gamma_1, ..., \gamma_n)$ taką, że $\gamma(i) = 0$, $i = 1, 2, ..., n$.

Krok 5. Wylosuj liczbę r z przedziału (0, 1). Znajdź punkt $C = (c_1, c_2, ..., c_n)$ taki, że $\|AB\| = \|AC\| + \|CB\|$ oraz $\|AC\| = r \cdot \|AB\|$. Dla każdego $k \in O$ wykonuj krok 6.

Krok 6. Połóż $l := m(k)$. Znajdź $t = \min_{i \in W_l} |c_{\varphi(k)} - i|$. Wylosuj element z ze zbioru $\{i \in W_l : |c_{\varphi(k)} - i| = t\}$. Połóż $\gamma(\varphi(k)) := z$ oraz $W_l := W_l \setminus \{z\}$.

Krok 7. Utwórz permutację δ taką, że $\delta^{-1} = \gamma$.

Krok 8. Połóż $(\varpi, C_{\max}(\varpi)) := P(PWK(\delta))$. Jeżeli $C_{\max}(\varpi) < C^*$, to połóż $\pi^* := \varpi$, $C^* := C_{\max}(\varpi)$.

Krok 9. Znajdź punkt $A \in R^n$ odpowiadający permutacji ϖ. Wylosuj liczbę r z przedziału (0, 1). Jeżeli $r \leq 0{,}5$, to połóż $B := A'$. W przeciwnym przypadku połóż $B := B'$.

Krok 10. Połóż $iter := iter + 1$. Jeżeli $iter < \max iterX$, to idź do kroku 4. W przeciwnym przypadku zwróć permutację π^*, wartość funkcji celu C^* i STOP.

Operator *MX* rozpoczyna działanie z ustalonymi permutacjami π, σ i parametrem *maxiterX*. W kroku 1 procedury tworzone są punkty $A = (a_1, ..., a_n)$, $B = (b_1, ..., b_n)$ odpowiadające permutacjom π, σ skonstruowane w myśl zasady przedstawionej w punkcie 3. W krokach 2, 3 tworzony jest podział $\theta = (O_1, ..., O_m)$, wykorzystywany przy konstrukcji permutacji potomnej. W kroku 4 tworzony jest zbiór $W = (W_1, W_2, ..., W_m)$, gdzie W_l, $l \in M$, jest zbiorem pozycji, na których mogą się znajdować operacje wykonywane na maszynie l w permutacji odwrotnej γ. W kroku 5 wyznaczany jest punkt C leżący na odcinku pomiędzy punktami A, B, oddalony o r jednostek od punktu A. W kroku 6 na podstawie punktu C

konstruowana jest permutacja odwrotna γ poprzez odpowiedni wybór elementów ze zbioru W. Permutacja δ, powstała w kroku 7, może być permutacją niedopuszczalną. Aby przywrócić jej cechy dopuszczalności, w kroku 8 został użyty algorytm priorytetowy (patrz. np. [4]) ze specyficzną regułą priorytetową $PWK(\delta)$. Zgodnie z tą regułą najwyższy priorytet ma operacja znajdująca się najbliżej początku permutacji δ_l, $l \in M$. Bardziej precyzyjnie, niech OK_l będzie zbiorem operacji konfliktowych na maszynie $l \in M$. Wtedy, zgodnie z powyższą regułą, najwyższy priorytet w tym zbiorze przyjmuje operacja i należąca do jednoelementowego zbioru $\{j \in OK_l : \delta^{-1}(j) = \min_{k \in OK_l} \delta^{-1}(k)\}$. Zapis $(\varpi, C_{\max}(\varpi)) := P(PWK(\delta))$ oznacza tutaj zastosowanie algorytmu priorytetowego z regułą $PWK(\delta)$ do permutacji δ. W wyniku działania algorytmu konstruowana jest permutacja dopuszczalna ϖ z wartością funkcji celu $C_{\max}(\varpi)$. Operator zwraca najlepszą permutację dopuszczalną π^*, odnalezioną w trakcie *maxiterX* iteracji kroków 4–10. Warto podkreślić, że poza permutacjami π, σ, parametr *maxiterX* jest jedynym parametrem wymaganym przez operator *MX*.

Przedstawiony wyżej operator krzyżowania został osadzony w schemacie algorytmu genetycznego, zaproponowanego przez Reevesa i Yamadę [3]. Dla zachowania ciągłości wywodu nowo powstały algorytm *GAMX* zostanie przedstawiony poniżej w całości.

Schemat algorytmu *GAMX*

Algorytm rozpoczyna działanie z parametrem *maxp*, parametrem *maxiterX* (dla operatora *MX*) i parametrami *maxiterTS*, *maxl*, *max*δ, *maxc*, *max_s*, *max_i*, *maxT* (dla algorytmu $TSAM_{AGV}$). Algorytm zwraca najlepsze znalezione rozwiązanie dopuszczalne i jego wartość funkcji celu.

Krok 1. Generuj populację $F = \{\pi^1, \pi^1, ..., \pi^{maxp}\}$, kładąc $(\pi^i, C_{\max}(\pi^i)) := P(RAN)$.

Krok 2. Uszereguj osobniki na liście F w ten sposób, by $C_{\max}(\pi^i) \le C_{\max}(\pi^{i+1})$, $i = 1, 2, ..., maxp - 1$.

Krok 3. Wylosuj dwa osobniki π^i, π^j, $i \neq j$, z prawdopodobieństwem odwrotnie proporcjonalnym do ich pozycji i, j na liście F.

Krok 4. Połóż $(\gamma, C_{\max}(\gamma)) := MX(\pi^i, \pi^j)$.

Krok 5. Połóż $(\delta, C_{\max}(\delta)) := TSAM_{AGV}(\gamma)$.

Krok 6. Jeżeli $C_{\max}(\delta) < C_{\max}(\pi^{maxp})$ oraz $C_{\max}(\delta) \neq C_{\max}(\pi^i)$, $i = 1, 2, ..., maxp - 1$, to połóż $\pi^{maxp} := \delta$.

Krok 7. Testuj kryterium stopu. Jeżeli kryterium stopu nie jest spełnione, to idź do kroku 2. W przeciwnym przypadku zwróć najlepsze rozwiązanie na liście F, jego wartość funkcji celu i STOP.

Za kryterium stopu algorytmu (w kroku 7) można przyjąć ograniczenie czasowe. Pewnego komentarza wymagają kroki 1, 4, 5. W kroku 1 populację F wygenerowano w sposób losowy za pomocą algorytmu priorytetowego. W kroku 4 przez zapis $(\delta, C_{\max}(\delta)) := MX(\pi, \sigma)$ należy rozumieć zastosowanie operatora *MX* do permutacji π, σ, gdzie δ jest permutacją dopuszczalną, powstałą w wyniku działania operatora. W kroku 5 do poprawy permutacji γ użyto algorytmu poszukiwań z zabronieniami $TSAM_{AGV}$ opisanego w pracy [1]. Algorytm *GAMX* może być zatem postrzegany jako schemat dywersyfikacji obliczeń dla algorytmu

$TSAM_{AGV}$. Algorytm $TSAM_{AGV}$ wymaga określenia szeregu parametrów, takich jak *maxiterTS* – maksymalna liczba iteracji bez poprawy najlepszej wartości funkcji celu, *maxl* – maksymalna długość listy dla skoków powrotnych, *max*δ i *maxc* – parametry używane w detektorze cykli, *max_s*, *max_i* – maksymalna długość list tabu, czy ograniczenie czasu pracy algorytmu – *maxT*. Poprzez zapis $(\delta, C_{\max}(\delta)) := TSAM_{AGV}(\gamma)$ należy rozumieć zastosowanie algorytmu $TSAM_{AGV}$ do permutacji γ, gdzie δ jest zwróconą permutacją.

5. Wyniki badań eksperymentalnych

Celem przeprowadzonych badań numerycznych było sprawdzenie, czy i w jakim stopniu algorytm *GAMX* poprawia rozwiązania dostarczone przez algorytm $TSAM_{AGV}$. W trakcie badań wykorzystano zestaw instancji testowych, które można opisać następująco. Instancje te zostały wygenerowane na bazie 40 instancji testowych Lawrence'a (LA01-LA40) dla klasycznego problemu gniazdowego, prezentowanych w pracy [5]. Instancje Lawrence'a są podzielone na 8 zestawów po 5 instancji o rozmiarach $r \times m^p$, 10×5, 15×5, 20×5, 10×10, 15×10, 20×10, 30×10, 15×15. Poprzez uwzględnienie różnych układów maszyn produkcyjnych, dodanie różnej liczby wózków AGV, różnych kombinacji czasów transportu oraz przejazdów pustych zostało wygenerowane 480 instancji podzielonych na 40 zestawów oznaczanych przez TM01, TM02, ..., TM40, po 12 instancji w każdym zestawie. Każda instancja ma swoją unikalną nazwę w formacie TM*a* / m^t / *b* / *c* / *d*, gdzie *a*, *b*, *c*, *d* są zmiennymi, zaś $m^t \in \{2, 4\}$ określa liczbę wózków AGV. Zmienna $a \in \{1, ..., 40\}$ oznacza numer instancji Lawrence'a, która podlega modyfikacji. Bardziej precyzyjnie, dla każdej instancji z grupy TM*a* przyjmuje się liczbę maszyn produkcyjnych, liczbę zadań i operacji produkcyjnych, czasy wykonania oraz przydział maszyn do operacji produkcyjnych taki sam jak dla instancji LA*a*. Zmienna $b \in \{1, 2\}$ określa typ układu maszyn produkcyjnych. Rozpatrywane były dwa, spośród najbardziej popularnych układów maszyn. Dla $b = 1$ oraz $b = 2$ przyjęto odpowiednio układ typu pętla oraz układ typu siatka. Zmienne *c* i *d* są współczynnikami skalującymi odpowiednio czasów przejazdów pustych i czasów transportu; mogą przyjmować jedną z trzech par wartości, $(c, d) \in \{(2, 2), (2, 5), (5, 5)\}$. Ostatecznie, dla każdej pary maszyn produkcyjnych $l_1, l_2 \in M^p$, przyjmuje się $e(l_1, l_2) = c \cdot g(l_1, l_2)$ oraz $t_k(l_1, l_2) = d \cdot g(l_1, l_2)$, $k \in J$, gdzie $g(l_1, l_2)$ jest funkcją określającą odległość pomiędzy maszynami l_1, l_2. Dla $b = 1$ przyjmuje się:

$$g(l_1, l_2) = \begin{cases} |l_1 - l_2|, & |l_1 - l_2| \le \left\lfloor \dfrac{m^p}{2} \right\rfloor \\ m^p - |l_1 - l_2|, & \text{w przeciwnym przypadku} \end{cases} \tag{2}$$

Dla $b = 2$ funkcja $g(l_1, l_2)$ przyjmuje postać:

$$g(l_1, l_2) = |t_1 - t_2| + \left| l_1 - t_1 - l_2 + \frac{t_2}{5} \right| \tag{3}$$

gdzie:

$$t_i = 5 \cdot \left(\frac{l_i - 1}{5} - \left\lfloor \frac{l_i - 1}{5} \right\rfloor \right) + 1, \quad i = 1, 2 \tag{4}$$

Dla każdej operacji transportowej $i \in O^t$ określony jest dwuelementowy zbiór $\mu(i) = \{a, a+1\}$, gdzie:

$$a = \min \left\{ m^t - 1, 1 + \left\lfloor \frac{(z_i - 1) \cdot (m^t - 1)}{r} \right\rfloor \right\} \tag{5}$$

zaś z_i jest numerem zadania, w ramach którego wykonywana jest operacja transportowa $i \in O^t$.

Spośród wszystkich zestawów instancji najbardziej interesujące wydają się zestawy TR16-25, oraz TR36-40 (w sumie 180 instancji) i głównie do nich będą ograniczone dalsze rozważania. Powodem tego zainteresowania jest stosunkowo niewielka liczba instancji rozwiązanych optymalnie w tych zestawach.

Wszystkie instancje testowe zostały rozwiązane w sposób przybliżony przez algorytmy zaimplementowane w Delphi 6 i uruchamiane na komputerze z procesorem AMD Athlon XP 2500+ (1833 MHz). Algorytm $TSAM_{AGV}$ dla każdej instancji TM został uruchomiony dwukrotnie z identycznymi wartościami parametrów jak w cytowanej pracy [1], tj. przyjęto *maxl* = 4, *max*δ = 100, *maxc* = 2, *max_s* = 15, *max_i* = 5 oraz *maxT* = 10 minut. W pierwszym uruchomieniu przyjęto *maxiterTS* = 16 000. W drugim uruchomieniu przyjęto *maxiterTS* = 30 000 w momencie startu algorytmu i po każdej poprawie wartości funkcji celu oraz *maxiterTS* = 10 000 po wykonaniu skoku powrotnego. Przez C^{TS} będzie oznaczana wartość funkcji celu najlepszego rozwiązania znalezionego przez algorytm dla każdej badanej instancji. Sumaryczny czas pracy algorytmu dla każdej instancji będzie oznaczany symbolem T^{TS}. Algorytm *GAMX* został uruchomiony dla każdej instancji sześciokrotnie z ograniczeniem czasowym (kryterium stopu) wynoszącym odpowiednio 200, 400,..., 1200 sekund oraz ustalonymi wartościami parametrów *maxl* = 4, *max*δ = 100, *maxc* = 2, *maxiterTS* = 16 000 oraz *maxT* = 20 minut. Wartości parametrów *max_s*, *max_i* przed każdym wywołaniem algorytmu $TSAM_{AGV}$ były losowane odpowiednio ze zbiorów {10, 11, ..., 20}, {3, 4, ..., 7}. W celu uproszczenia notacji, algorytm z ograniczeniem czasowym 200, 400, ..., 1200 sekund będzie nazywany odpowiednio algorytmem *GAMX*1, *GAMX*2, ..., *GAMX*6, zaś zwrócone przez niego wartości funkcji celu będą oznaczane odpowiednio przez C^{MX1}, C^{MX2}, ..., C^{MX6}.

Na podstawie wyznaczonych wielkości C^A, $A \in \{MX1, MX2, ..., MX6\}$, zostały obliczone procentowe poprawy $\Phi_A^{TS} = 100\% \cdot (C^{TS} - C^A)/C^{TS}$ wartości C^{TS} przez algorytm genetyczny. Dla każdego zestawu TM16-25, TM36-40 wyliczono średnie arytmetyczne $av\Phi_A^{TS}$ popraw oraz średnie arytmetyczne avT^{TS} sumarycznego czasu pracy algorytmu $TSAM_{AGV}$ i przedstawiono w tabeli 1. W tabeli 2 naniesione są wartości funkcji celu najlepszych rozwiązań znalezionych przez wszystkie algorytmy; jeżeli zachodzi $C^{MX6} < C^{TS}$ albo $C^{MX6} = C^{TS}$, wartości C^{MX6} zostały odpowiednio pogrubione albo oznaczone kursywą.

Tabela 1
Średnie arytmetyczne $av\Phi_A^{TS}$ popraw i czasu pracy avT^{TS} algorytmu $TSAM_{AGV}$

TM*a*	$av\Phi_{MX1}^{TS}$ [%]	$av\Phi_{MX2}^{TS}$ [%]	$av\Phi_{MX3}^{TS}$ [%]	$av\Phi_{MX4}^{TS}$ [%]	$av\Phi_{MX5}^{TS}$ [%]	$av\Phi_{MX6}^{TS}$ [%]	avT^{TS} [s]
TM16	0,36	0,43	0,44	0,44	0,44	0,48	37,2
TM17	0,07	0,09	0,09	0,09	0,09	0,09	76,2
TM18	0,46	0,50	0,50	0,50	0,54	0,54	54,4
TM19	0,36	0,45	0,51	0,51	0,52	0,53	66,4
TM20	0,12	0,15	0,16	0,16	0,16	0,18	43,5
TM21	–0,20	–0,17	–0,15	0,01	0,07	0,08	191,0
TM22	–0,53	–0,44	–0,34	–0,28	–0,18	–0,18	282,1
TM23	–0,32	–0,27	–0,13	–0,10	–0,08	–0,07	243,8
TM24	0,47	0,58	0,59	0,65	0,70	0,74	335,8
TM25	–0,22	0,02	0,08	0,16	0,18	0,23	220,8
TM36	–1,23	–0,62	–0,55	–0,51	–0,37	–0,30	704,6
TM37	–1,48	–0,93	–0,47	–0,31	–0,28	–0,28	602,4
TM38	–2,44	–1,96	–1,80	–1,66	–1,57	–1,54	796,8
TM39	–1,37	–1,14	–0,70	–0,62	–0,50	–0,44	691,0
TM40	–1,27	–0,91	–0,76	–0,70	–0,54	–0,46	748,0
Średnia	**–0,48**	**–0,28**	**–0,17**	**–0,11**	**–0,05**	**–0,03**	**339,6**

Tabela 2
Wartości funkcji celu najlepszych rozwiązań dla instancji z zestawów TM 16-25, TM 36-40

TMa	2/1/c/d			2/2/c/d			4/1/c/d			4/2/c/d		
	2/2	2/5	5/5	2/2	2/5	5/5	2/2	2/5	5/5	2/2	2/5	5/5
TM16	*976*	**1040**	**1065**	**976**	**1017**	**1021**	*976*	**1040**	*1040*	**976**	*1017*	*1017*
TM17	*805*	866	923	*801*	**830**	857	*805*	*848*	*848*	*801*	*830*	*830*
TM18	**881**	**931**	**966**	**874**	**919**	**925**	**877**	*927*	*927*	**874**	**913**	**913**
TM19	*874*	**920**	**954**	**876**	**920**	**946**	*874*	*918*	*918*	*876*	*920*	*920*
TM20	**931**	*978*	**986**	*934*	*967*	**969**	*931*	*975*	*975*	*934*	*967*	*967*
TM21	**1082**	**1153**	1224	**1081**	**1157**	1212	**1082**	*1124*	**1126**	*1081*	**1123**	**1123**
TM22	*953*	1078	1172	**947**	**1017**	1099	*953*	**988**	*989*	*947*	**976**	**977**
TM23	*1032*	**1130**	1213	*1032*	1065	**1153**	*1032*	**1041**	**1044**	*1032*	**1035**	**1039**
TM24	**967**	**1149**	1246	**965**	**1063**	1164	**966**	**1022**	1021	**963**	**1007**	**1008**
TM25	*1001*	**1110**	1210	**999**	**1082**	1146	**1001**	**1041**	**1041**	*998*	*1043*	*1043*
TM36	**1430**	2479	2759	1345	1761	2013	**1357**	1562	1720	*1331*	**1406**	**1444**
TM37	**1530**	2383	2690	**1482**	1731	1933	1501	**1645**	1741	**1472**	**1557**	*1561*
TM38	1391	2445	2751	**1298**	1635	1888	**1313**	1498	1641	**1263**	1337	1381
TM39	**1378**	2268	2516	1279	1669	1887	1308	**1493**	1596	**1274**	**1365**	1405
TM40	**1364**	2351	2642	**1286**	1679	1916	1293	1471	1605	1268	**1355**	**1393**

Jak widać, dla instancji z zestawów TM16-20 algorytm *GAMX* generuje rozwiązania nie gorsze niż algorytm $TSAM_{AGV}$ już w stosunkowo krótkim horyzoncie czasowym. Algorytm *GAMX*6 poprawił wartość C^{TS} dla 26 instancji. Tylko 3 wartości C^{MX6} są mniejsze od odpowiednich wartości C^{TS}. W tabeli 1 można też zaobserwować, że wydłużenie czasu pracy algorytmu z 200 do 1200 sekund nie przynosi istotnej poprawy jakości generowanych rozwiązań. W przypadku zestawów TM21-25 algorytm genetyczny nadal generuje rozwiązania średnio lepsze niż porównywany algorytm. Wygenerowanie takich rozwiązań wiąże się jednak ze znacznie większym nakładem obliczeniowym niż w przypadku instancji z grup TM16-20. Wartość C^{MX6} jest większa od wartości C^{TS} w 34 przypadkach, zaś mniejsza dla 12 instancji. W przypadku instancji z zestawów TM36-40 algorytm *GAMX*, pomimo nielicznych wyjątków, wygenerował rozwiązania gorszej jakości, niż algorytm poszukiwań z zabronieniami, niezależnie od czasu pracy. Wartość C^{MX6} jest większa od wartości C^{TS} jedynie dla 20 instancji, podczas gdy niemal we wszystkich pozostałych przypadkach jest gorsza.

Podsumowując, można stwierdzić, że przeprowadzone badania numeryczne wykazały, iż nawet zwielokrotnienie liczby uruchomień algorytmu $TSAM_{AGV}$ nie gwarantuje znacznej poprawy jakości generowanych przez niego rozwiązań. Z drugiej jednak strony algorytm *GAMX*6 poprawił 80 spośród 180 rozwiązań wygenerowanych przez algorytm 2, zaś gorszy okazał się w 53 przypadkach. Opisywany algorytm genetyczny można zatem traktować jako dobre uzupełnienie samodzielnego algorytmu poszukiwań z zabronieniami.

6. Zakończenie

W pracy został przedstawiony nowy quasi-operator krzyżowania *MX* oraz hybrydowy algorytm genetyczny *GAMX*. Algorytm ten wykorzystuje w trakcie swojej pracy, między innymi, algorytm poszukiwań z zabronieniami $TSAM_{AGV}$. Algorytm został użyty do przybliżonego rozwiązania problemu gniazdowego z transportem i ograniczoną liczbą niededykowanych wózków AGV. Dostarczone rozwiązania zostały porównane z rozwiązaniami dostarczonymi bezpośrednio przez algorytm $TSAM_{AGV}$ (uruchamianym jednak ze znacznie większym limitem czasowym, niż miało to miejsce w przypadku algorytmu *GAMX*). Przeprowadzone badania numeryczne ujawniły, że stosunkowa jakość rozwiązań generowanych przez algorytm *GAMX* bardzo silnie zależy od poszczególnych grup rozwiązywanych instancji.

Literatura

[1] Smutnicki C., Tyński A.: *Problem gniazdowy z transportem i ograniczoną liczbą niededykowanych wózków AGV.* Automatyka, 2005, 9, 233

[2] Smutnicki C., Tyński A.: *Job-Shop Scheduling by GA. A New Crossover Operator.* Operations Research, 2006, 715

[3] Reeves C., Yamada T.: *Genetic Algorithms, Path Relinking and the Flowshop Sequencing Problem. Evolutionary Computation.* 1995, 80, 397
[4] Grabowski J., Nowicki E., Smutnicki C.: *Metoda blokowa w zagadnieniach szeregowania zadań.* Warszawa, EXIT 2003
[5] Lawrence S.: *Resource Constrained Project Scheduling: an Experimental Investigation of Heuristic Scheduling Techniques.* Pitsburg, Graduate School of Industrial Administration. Carneige-Mellon University 1984