Raport Badawczy
Research Report

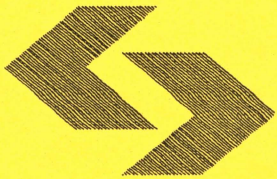
RB/13/2002

Wybrane algorytmy rozwiązania problemu szeregowania zadań z uwzględnieniem ruchu realizatorów

J. Józefczyk

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences



POLSKA AKADEMIA NAUK

Instytut Badań Systemowych

ul. Newelska 6

01-447 Warszawa

tel.: (+48) (22) 8373578

fax: (+48) (22) 8372772

Kierownik Pracowni zgłaszający pracę:
Prof. dr hab. inż. Zdzisław Bubnicki

Warszawa 2002

WYBRANE ALGORYTMY ROZWIĄZANIA PROBLEMU SZEREGOWANIA ZADAŃ Z UWZGLĘDNIENIEM RUCHU REALIZATORÓW

Jerzy Józefczyk

Instytut Badań Systemowych PAN

Scheduling of tasks on moving executors is considered. It is formulated as the discrete optimization problem. The overview of solution algorithms is given. Approximate and AI-based algorithms are presented. In the approximate algorithm the solution algorithms for solving the classical scheduling problem and TSP are applied. Different versions of the evolutionary algoritm are also described. They comprise adaptation or learning of crossover and mutation probabilities. The comparison of the scheduling algorithms presented via computer simulation is given.

Key words: scheduling algorithms, operations research, optimization problems, moving objects, evolutionary algorithms

1. Wstęp

Problematyka szeregowania zadań jest aktualna i intensywnie rozwijana od szeregu lat. Nowe kierunki badawcze są inspirowane przez rozwój środków informatyki a w konsekwencji przez wzrost możliwości efektywnej realizacji opracowywanych algorytmów szeregowania. Przede wszystkim jednak badania w zakresie szeregowania zadań są kierunkowane przez zapotrzebowanie na algorytmy szeregowania w najważniejszych obszarach ich zastosowań, tj. w informatycznych systemach obliczeniowo-decyzyjnych oraz w dyskretnych systemach produkcyjnych. Spośród wielu ciekawych i nowoczesnych kierunków badawczych warto zwrócić uwagę na zagadnienia szeregowania zadań na wielu realizatorach, problemy szeregowania zadań w warunkach niepewności, czyli przede wszystkim z niezdeterminowanymi czasami

wykonania zadań oraz zagadnienia szeregowania z nieznanymi a priori czasami wykonania zadań, np. (Browne (1990), Chen (1996), Janiak (1998)). Wymienione kierunki badawcze skupiające szereg szczegółowych problemów szeregowania umożliwiają stosowanie opracowanych algorytmów szeregowania w takich systemach praktycznych, w których użycie klasycznych algorytmów szeregowania było niemożliwe. Inną wspólną cechą wymienionych kierunków badawczych są próby stosowania tzw. metod sztucznej inteligencji do wyznaczania algorytmów szeregowania – przede wszystkim jako algorytmów przybliżonych dla problemów NP-trudnych lub silnie NP-trudnych. Warto tu wspomnieć o wykorzystywaniu formalizmów opartych na zbiorach rozmytych i podejść stosujących wybraną logikę wielowartościową w problemach niedeterministycznych (Medskev (1995)), a także o rozwiązywaniu określonych problemów szeregowania traktowanych jako problemy optymalizacji dyskretnej przy użyciu technik ewolucyjnych (Baeck (1996), Foncesca (1995), Srinivas *et al* (1994)).

Prezentowana praca nawiązuje do zarysowanych zagadnień badawczych zarówno w sensie podejmowania aktualnych problemów metodologicznych, jak i z powodu stosowania nowoczesnych technik wyznaczania algorytmów rozwiązania. Punktem odniesienia do rozważań jest dyskretny system produkcyjny składający się m.in. z obiektów, na których są wykonywane zadania technologiczne (zadania), ze stanowisk, czyli miejsc na których te zadania są wykonywane oraz z realizatorów jako podmiotów wykonujących zadania. Jest oczywiste, że do wykonania określonego zestawu zadań, a w konsekwencji do wykonania obiektów niezbędny jest ruch niektórych elementów systemu produkcyjnego. W tradycyjnych rozwiązaniach ruchome są obiekty. Przemieszczają się one miedzy stanowiskami produkcyjnymi, na których są zlokalizowane nieruchome realizatory. W pracy przyjęto mniej konwencjonalne, ale spotykane w praktycznych systemach produkcyjnych i usługowych założenie, że nieruchome są obiekty umieszczone na stanowiskach, a poruszają się realizatory. Założenie to jest podstawą sformułowania oryginalnych problemów szeregowania, które można zaliczyć do

trzeciego z wymienionych kierunków badawczych, ponieważ, jak się okazuje, nie można wówczas przyjąć, że czasy wykonania zadań są a priori dane. Taki problem badawczy – określany jako problem *szeregowania z uwzględnieniem ruchu realizatorów* – był i jest opracowywany, a niektóre rezultaty są przedstawione w pracach (Józefczyk (1996), (1997), (2001a), (2001c)). Rozważano kilka szczegółowych przypadków, np. dla kryteriów w postaci długości uszeregowania i maksymalnego opóźnienia. W obu przypadkach dla zadań niepodzielnych (a tylko takie były rozważane dla rozpatrywanych systemów produkcyjnych) odpowiednie problemy szeregowania są NP-trudnymi problemami optymalizacyjnymi. W związku z tym kluczową sprawą jest wyznaczanie efektywnych algorytmów przybliżonych. W pracy jest przedstawiony krótki przegląd i porównanie uzyskanych dotąd algorytmów rozwiązania dla wybranego zagadnienia szeregowania z kryterium w postaci długości uszeregowania. Algorytmy te nawiązują do tradycyjnych metod jak i do metod opartych na podejściu ewolucyjnym.

W rozdziale 2. wprowadzono i sformułowano rozpatrywany w pracy problem szeregowania z ruchomymi realizatorami. Następnie w rozdziale 3. krótko scharakteryzowano sposób uzyskiwania algorytmu dokładnego wykorzystujący zasadę podziału i ograniczeń, a także przedstawiono algorytm przybliżony uzyskany po dokonaniu tzw. funkcjonalnej dekompozycji problemu szeregowania. W kolejnym rozdziale omówiono zastosowanie algorytmu ewolucyjnego do rozwiązania problemu. Przedstawiono trzy jego wersje, w których w różny sposób ustalano podstawowe parametry algorytmu. Rozważania zakończono podaniem wybranych wyników badań symulacyjnych mających na celu porównanie algorytmów rozwiązania.

2. Sformułowanie problemu szeregowania z uwzględnieniem ruchu realizatorów

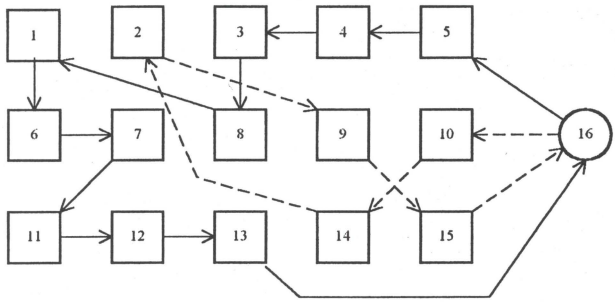
Dla wprowadzonego we wstępie problemu szeregowania, w którym zadania są wykonywane na stanowiskach przez ruchome realizatory, rozważamy problem szeregowania H zadań tworzących zbiór $\boldsymbol{H} = \{1,2,...,H\}$. Przyjmujemy, że każde zadanie $h \in \boldsymbol{H}$ jest wykonywane na innym stanowisku oraz, że na każdym stanowisku jest wykonywane zadanie. W

konsekwencji zbiory zadań i stanowisk są tożsame. Dodatkowo wyróżniamy stanowisko $h = H + 1$ będące miejscem, w którym każdy realizator rozpoczyna i kończy pracę. Wówczas $\overline{H} = \boldsymbol{H} \cup \{H + 1\}$ jest zbiorem stanowisk wraz z bazą. Zbiór realizatorów oraz ich liczbę oznaczamy odpowiednio jako $\boldsymbol{R}$ oraz R. Każde zadanie składa się z dwóch części: z dojazdu realizatora do stanowiska z innego stanowiska lub z bazy oraz z wykonania czynności na stanowisku. Stąd czas $\tau_{r,h}$ wykonania bieżącego zadania h przez bieżący realizator r jest sumą odpowiedniego czasu dojazdu $\hat{\tau}_{r,g,h}$ oraz czasu wykonania czynności na stanowisku h (czynności w ramach zadania h) $\bar{\tau}_{r,h}$, czyli

$$\tau_{r,h} = \bar{\tau}_{r,h} + \hat{\tau}_{r,g,h},\ r = 1,2,...,R,\ h = 1,2,...,H,\ g = 1,2,...,H+1.$$

Przyjęcie założenia, że po wykonaniu wszystkich przydzielonych zadań każdy realizator musi się znaleźć w bazie oznacza, że $(\forall r \in \boldsymbol{R})\,(\exists g \in \boldsymbol{H})\,\hat{\tau}_{r,g,H+1} > 0$. Zakładamy też, że $\bar{\tau}_{r,H+1} = 0$ dla $r = 1,2,...,R$, czyli że w bazie nie są wykonywane żadne czynności. W dalszym ciągu rozważamy realizatory dowolne. Oznacza to, że różnią się one prędkościami wykonywania zadań oraz prędkości te zależą od wykonywanego zadania, przy czym dowolne zadanie może być wykonywane przez każdy z dostępnych realizatorów. Precyzując rozważany dalej problem szeregowania ustalamy równe momenty gotowości dla wszystkich zadań, niepodzielność zadań co dla zadań produkcyjnych jest typowe oraz brak ograniczeń kolejnościowych co jest już znacznym uproszczeniem, które umożliwia jednak czytelne ukazanie istoty problemu szeregowania z ruchomymi realizatorami – a zwłaszcza porównanie różnych algorytmów szeregowania, co jest głównym celem pracy. Jako kryterium jakości szeregowania będzie dalej przyjmowana długość uszeregowania. Celem szeregowania jest nie tylko wyznaczanie podzbiorów zadań do wykonania przez poszczególne realizatory – jak to ma miejsce w przypadku klasycznym – ale również kolejności ich wykonywania. Dlatego faktycznie dla każdego realizatora należy wyznaczyć trasę o początku i końcu w bazie, czyli tzw. cykl Ha-

miltona. Zbiór takich tras powinien mieć własność, że każde zadanie (stanowisko) powinno się znaleźć tylko w jednym cyklu. Przykładową strukturę rozpatrywanego systemu produkcyjnego dla $H = 15$ i $R = 2$ przedstawiono na rys. 1. Prostokątami oznaczono stanowiska (zadania), kółkiem – bazę, a linie ciągła i przerywana oznaczają trasy poruszania się odpowiednio pierwszego i drugiego realizatora. Rozważane zagadnienie szeregowania przedstawimy jako problem optymalizacji dyskretnej. W tym celu wprowadzamy trójwymiarową, binarną macierz decyzyjną $\gamma = [\gamma_{r,g,h}]_{\substack{r=1,2,...,R \\ g,h=1,2,...,H+1}}$



Rys.1. Przykładowa struktura systemu produkcyjnego

traktowaną jako zmienną optymalizacyjną, której bieżący element jest zdefiniowany następująco: $\gamma_{r,g,h} = 1(0)$ jeśli realizator r wykonuje zadanie h po dojeździe ze stanowiska g (w przeciwnym przypadku).

Następujące ograniczenia nałożone na macierz γ pozwalają na wyznaczanie rozwiązań dopuszczalnych (cykli Hamiltona)

$$\gamma_{r,g,h} \in \{0,1\}, \quad r = 1,2, ..., R, \quad g,h = 1,2, ..., H+1, \tag{1}$$

$$\gamma_{r,h,h} = 0, \quad r = 1,2, ..., R, \quad h = 1,2, ..., H+1, \tag{2}$$

$$\sum_{r=1}^{R} \sum_{g=1}^{H+1} \gamma_{r,g,h} = 1, \quad h = 1,2, ..., H, \tag{3}$$

$$\sum_{g=1}^{H+1} \gamma_{r,g,p} = \sum_{h=1}^{H+1} \gamma_{r,p,h},\ r = 1,2,...,R,\ p = 1,2,...,H+1, \tag{4}$$

$$\gamma \in \boldsymbol{S}, \tag{5}$$

gdzie $\boldsymbol{S} = \{\gamma : \sum_{g \in \overline{\boldsymbol{H}}_S} \sum_{h \in \overline{\boldsymbol{H}}_S} \gamma_{r,g,h} \le \overline{H}_S - 1,\ \overline{\boldsymbol{H}}_S$ – dowolny niepusty podzbiór $\overline{\boldsymbol{H}}$, $r = 1,2,...,R\}$

$$\sum_{h=1}^{H} \gamma_{r,H+1,h} = 1, r = 1,2,...,R. \tag{6}$$

Ograniczenia (1) i (2) są oczywiste. Następne zapewnia, że każde zadanie będzie wykonane. Kolejne dwa umożliwiają wyznaczenie tras przejazdów realizatorów tylko jako cykli Hamiltona. Ograniczenie (6) zapewnia, że każdy realizator dokładnie raz wyrusza z bazy. Długość uszeregowania możemy przedstawić w postaci następującego wyrażenia

$$Q(\gamma) = \max_{r=1,2,...,R} \left\{ \sum_{h=1}^{H+1} \sum_{g=1}^{H+1} \gamma_{r,g,h} (\overline{\tau}_{r,h} + \hat{\tau}_{r,g,h}) \right\}. \tag{7}$$

Szeregowanie zadań z ruchomymi realizatorami polega na wyznaczeniu macierzy γ dopuszczalnej w sensie ograniczeń (1)–(6) tak, aby minimalizować kryterium (7) – dla danych w postaci zbioru zadań (stanowisk) $\overline{\boldsymbol{H}}$, zbioru realizatorów $\boldsymbol{R}$ oraz macierzy czasów dojazdów $\hat{\tau} = [\hat{\tau}_{r,g,h}]_{\substack{r=1,2,...,R \\ g,h=1,2,...,H+1}}$ i macierzy czasów wykonywania czynności na stanowiskach $\overline{\tau} = [\overline{\tau}_{r,h}]_{\substack{r=1,2,...,R \\ h=1,2,...,H}}$. Problem ten będziemy w dalszym ciągu oznaczać jako PC.

3. Własności problemu i tradycyjne metody rozwiązania.

Sformułowany w poprzednim rozdziale problem szeregowania zadań z uwzględnieniem ruchu realizatorów (PC) jest NP-trudnym problemem optymalizacyjnym. Aby uzasadnić tę

własność, wystarczy zauważyć, że w szczególnym przypadku, jeśli dla każdego zadania i realizatora wszystkie czasy dojazdów są takie same, tzn. $\hat{\tau}_{r,g,h} = \check{\tau}_{r,h}$, $g = 1,2,...,H+1$, $g \neq h$ dla $r \in \boldsymbol{R}$ i $h \in \overline{\boldsymbol{H}}$, to wówczas czasy wykonania zadań $\tau_{r,h}$ są dane a priori i otrzymujemy klasyczny problem szeregowania zadań niepodzielnych i niezależnych na realizatorach dowolnych w celu minimalizacji długości uszeregowania, który jest NP-trudny.

3.1. Algorytm dokładny

Dokładny algorytm rozwiązania o złożoności wykładniczej został przedstawiony w pracy Józefczyk (1996). Wykorzystuje on tzw. dekompozycję czasową problemu sformułowanego w rozdz. 2. Zasada tej dekompozycji polega na podziale horyzontu czasowego, w którym są wykonywane wszystkie zadania ze zbioru $\boldsymbol{H}$ oraz zjazdy realizatorów do bazy, na krótsze przedziały czasu i na podejmowaniu decyzji o wykonywaniu zadań i ruchu realizatorów w takich przedziałach, z uwzględnieniem skutków decyzji podjętych wcześniej. Można wykazać (Józefczyk (2000c)), że otrzymany w ten sposób problem decyzyjny jest równoważny problemowi PC. Algorytm ten jest skuteczny tylko dla niewielkich rozmiarów problemu i służy głównie jako podstawa do oceny innych, czyli przybliżonych lub heurystycznych algorytmów rozwiązania.

3.2. Algorytm przybliżony

Podstawowy przybliżony algorytm szeregowania wykorzystuje tzw. dekompozycję funkcjonalną. Procedura dekompozycji problemu PC na prostsze podproblemy optymalizacyjne polega na podziale trójwymiarowych macierzy γ na macierze dwuwymiarowe, będące warstwami w macierzy γ, odpowiadającymi różnym realizatorom ze zbioru $\boldsymbol{R}$. Idea takiej dekompozycji polega na tym, aby problem rozwiązywać osobno dla różnych realizatorów. Z prostej analizy ograniczeń nałożonych na γ można wywnioskować, że rozważane warstwy macierzy γ są ze sobą powiązane i bez straty optymalności nie można wyznaczać niezależnie elementów wspomnianych macierzy dwuwymiarowych. Nowe macierze decyzyjne wynikają-

ce z γ oznaczamy jako $\tilde{\gamma}^r = [\tilde{\gamma}^r_{g,h}]_{g,h=1,2,...,H+1}$, $r = 1,2,...,R$, gdzie $\gamma^r_{g,h} = 1(0)$ jeśli realizator r wykonuje zadanie h po dojeździe ze stanowiska g (w przeciwnym przypadku). Zawierają one informacje o trasach przejazdów poszczególnych realizatorów. Mogą one być traktowane jako niezależne warstwy macierzy γ. Na macierze $\tilde{\gamma}^r$ są nałożone cztery warunki w celu zapewnienia ich dopuszczalności, a mianowicie

$$\tilde{\gamma}^r_{h,h} = 0, \quad h = 1,2,...,H+1, \tag{8}$$

$$\sum_{g=1}^{H+1} \tilde{\gamma}^r_{g,h} \le 1, \quad h = 1,2,...,H+1, \tag{9}$$

$$\sum_{g=1}^{H+1} \tilde{\gamma}^r_{g,p} = \sum_{h=1}^{H+1} \tilde{\gamma}^r_{p,h}, \quad p = 1,2,...,H+1, \tag{10}$$

$$\tilde{\gamma}^r \in S_r, \tag{11}$$

gdzie $S_r = \left\{ \tilde{\gamma}^r : \sum_{g \in \overline{H}^r_S} \sum_{h \in \overline{H}^r_S} \tilde{\gamma}^r_{g,h} \le \overline{H}^r_S - 1 \right\}$ i $\overline{H}^r_S$ jest dowolnym niepustym podzbiorem $\overline{H}$.

W celu koordynacji macierzy $\tilde{\gamma}^r$ wprowadzamy nową macierz decyzyjną

$$c = [c_{r,h}]_{\substack{r=1,2,...,R \\ h=1,2,...,H+1}},$$

gdzie $c_{r,h} = 1(0)$, jeśli realizator r wykonuje zadanie h (w przeciwnym przypadku).

Zakładamy, że

$$\sum_{r=1}^{R} c_{r,h} = 1, \quad h = 1,2,...,H \tag{12}$$

oraz

$$c_{r,H+1} = 1, \quad r = 1,2,...,H. \tag{13}$$

Oznacza to, że każde zadanie musi być wykonane, oraz że każdy realizator musi dojechać do bazy. Okazuje się, że nowy problem optymalizacyjny powstały w wyniku zastąpienia

trójwymiarowej macierzy decyzyjnej γ zestawem dwuwymiarowych macierzy decyzyjnych tworzących $\tilde{\gamma} \triangleq [\tilde{\gamma}^1 | \tilde{\gamma}^2 | \cdots | \tilde{\gamma}^R]$ oraz macierzy $\boldsymbol{c}$, zgodnie z twierdzeniem, jest równoważny problemowi wyjściowemu PC.

Twierdzenie

Jeżeli związek między macierzami zmiennych decyzyjnych γ oraz $\boldsymbol{c}$ i $\tilde{\gamma}$ jest określony przez zależność $\gamma_{r,g,h} = c_{r,h} \cdot \tilde{\gamma}^r_{g,h}$, $g,h = 1,2,...,H+1$, $r =,2,...,R$, to problem wyjściowy PC jest równoważny w sensie kryteriów jakości oraz ograniczeń minimalizacji względem $\boldsymbol{c}$ i $\tilde{\gamma}$ kryterium jakości

$$\tilde{Q}_C(\boldsymbol{c},\tilde{\gamma}) = \max_{r=1,2,...,R} \left\{ \sum_{h=1}^{H+1} c_{r,h} \left(\bar{\tau}_{r,h} + \sum_{g=1}^{H+1} \tilde{\gamma}^r_{g,h} \cdot \hat{\tau}_{r,g,h} \right) \right\}$$

z ograniczeniami (12) i (13) na $\boldsymbol{c}$ oraz następującymi ograniczeniami na $\tilde{\gamma}$

$$\tilde{\gamma}^r_{h,h} = 0, \text{ dla } c_{r,h} = 1, \quad h = 1,2,...,H+1, \quad r = 1,2,...,R,$$

$$\sum_{g=1}^{H+1} \tilde{\gamma}^r_{g,h} = 1, \text{ dla } c_{r,h} = 1, \quad h = 1,2,...,H+1, \quad r = 1,2,...,R,$$

$$\sum_{g=1}^{H+1} \tilde{\gamma}^r_{g,p} = \sum_{h=1}^{H+1} \tilde{\gamma}^r_{p,h}, \quad p = 1,2,...,H+1, \text{ dla } c_{r,h} = 1, \quad h = 1,2,...,H+1, \quad r = 1,2,...,R,$$

$$\tilde{\gamma}^r \in \boldsymbol{S}_r, \quad r = 1,2,...,R,$$

gdzie zbiory $\boldsymbol{S}_r$ są określone w (11), przy czym dla występujących tam indeksów g i h jest prawdziwa równość $c_{r,g} = c_{r,h} = 1$.

Dowód podano w Józefczyk (1996). Istnienie powiązania między zmiennymi decyzyjnymi $\boldsymbol{c}$ oraz $\tilde{\gamma}^1, \tilde{\gamma}^2, ..., \tilde{\gamma}^R$ w formie relacji uniemożliwia zmniejszenie wymiarowości problemu optymalizacyjnego uzyskane w wyniku dekompozycji, a polegające na oddzielnym wyznaczaniu macierzy $\boldsymbol{c}$ oraz $\tilde{\gamma}^1, \tilde{\gamma}^2, ..., \tilde{\gamma}^R$. Dzieje się tak, ponieważ w celu wyznaczenia $\boldsymbol{c}$

są potrzebne pewne informacje zawarte w $\tilde{\gamma}^r, r = 1,2, ..., R$. Nie jest konieczna znajomość $\tilde{\gamma}^r$, ponieważ $\tilde{\gamma}^r$ jednoznacznie określa $\boldsymbol{c}$, ale jest potrzebna informacja o stanowisku, z którego realizator podąża w celu wykonania bieżącego zadania. Traktując sprawę formalnie, mówimy o takiej sytuacji, w której spośród wszystkich $\tilde{\gamma}^r_{g,h}$ dla $g = 1,2,...,H+1$ dokładnie jeden element ma wartość równą 1. Własność taką mają macierze, dla których jest spełniony warunek $\sum_{g=1}^{H+1} \tilde{\gamma}^r_{g,h} = 1, h = 1,2, ..., H+1$. Proponowany algorytm rozwiązania polega na wstępnym ustaleniu początkowych wartości macierzy $\tilde{\gamma}^r = \tilde{\gamma}^r_0$, a następnie na kolejnym wyznaczaniu $\boldsymbol{c}$ oraz $\tilde{\gamma}^r$. Można go przedstawić w formie trzech następujących kroków

1. Ustalamy $\tilde{\gamma}^r = \tilde{\gamma}^r_0$.

2. Rozwiązujemy problem szeregowania polegający na minimalizacji, względem $\boldsymbol{c}$ dopuszczalnych w sensie (12) i (13), wyrażenia

$$\max_{r=1,2,...,R} \left\{ \sum_{h=1}^{H} c_{r,h} \cdot \left(\bar{\tau}_{r,h} + \sum_{g=1}^{H+1} \tilde{\gamma}^r_{0,g,h} \cdot \hat{\tau}_{r,g,h} \right) \right\}.$$

W rezultacie otrzymujemy rozłączne zbiory $\overline{H}^{r,*}_c$, $r = 1,2,..., R$ zadań wykonywane przez poszczególne realizatory.

3. Rozwiązujemy R niezależnych problemów komiwojażera dla zadań ze zbiorów $\overline{H}^{r,*}_c$ (osobno dla każdego realizatora) i otrzymujemy trasy przejazdów realizatorów o początku i końcu w bazie.

Do rozwiązania problemów optymalizacyjnych z kroków 2. i 3. można stosować dowolne (dokładne, przybliżone lub heurystyczne) szczegółowe algorytmy rozwiązujące odpowiednie zagadnienia szeregowania i komiwojażera. W pracy Józefczyk (1996) przedstawiono szczegółową analizę tego algorytmu, w szczególności zbadano jego własności w sposób ana-

lityczny i z wykorzystaniem metod symulacji komputerowej.

4. Zastosowanie algorytmów ewolucyjnych

Analiza własności algorytmu przybliżonego przedstawionego w p.3.2 wskazała na potrzebę poszukiwania nowych narzędzi przydatnych do wyznaczania rozwiązań rozważanego złożonego problemu szeregowania. Zdecydowano się wykorzystać w tym celu algorytmy ewolucyjne, będące skutecznym i coraz częściej stosowanym ostatnio narzędziem do wyznaczania optimów globalnych różnych problemów optymalizacyjnych – również z zakresu optymalizacji dyskretnej. W dalszym ciągu przedstawimy trzy wersje algorytmu szeregowania, wszystkie oparte na podejściu ewolucyjnym. Wprowadzimy wpierw podstawowe pojęcia i oznaczenia właściwe dla algorytmów ewolucyjnych. Rozwiązanie dopuszczalne, będące w rozważanym przypadku trasami poszczególnych realizatorów, jest zapisywane w formie ciągu genów $(g_1, g_2, ..., g_M)$ o długości $M = H + R$. Ciąg ten jest podzielony na dwie części. Pierwsza składa się z H, a druga z R elementów. W drugiej części zapisano liczby zadań wykonywanych przez realizatory r, $r = 1,2, ..., R$, czyli $g_{H+r} = M_r$. W pierwszej części kolejno dla poszczególnych realizatorów są zakodowane trasy przejazdów realizatorów, w postaci numerów zadań wykonywanych przez realizatory. Bieżący element $m_r(j)$, $j = 1,2, ..., M_r$ trasy $\boldsymbol{M}_r$ r-tego realizatora, będący numerem odpowiedniego stanowiska (zadania) jest zapisany jako element g_m, gdzie $m = j + \sum_{i=0}^{r-1} M_i$ oraz $M_0 = 0$. Trasy $\boldsymbol{M}_r$ zawierają ciągi zadań wykonywanych przez realizatory, uzupełnione przez bazę $H + 1 = 21$, stanowiącą początek i koniec każdej trasy. Efektywność algorytmów ewolucyjnych polegająca na osiąganiu ekstremów globalnych funkcji, silnie zależy od wartości parametrów tych algorytmów takich jak : liczność populacji I, liczba iteracji (liczba pokoleń) J, a przede wszystkim prawdopodobieństwo krzyżowania p_1 i prawdopodobieństwo mutacji p_2. Zastosowanie podstawowego schematu algorytmu genetycznego polega na uruchamianiu w kolejnych iteracjach podsta-

wowych operatorów genetycznych, tzn. selekcji, krzyżowania i mutacji. Do selekcji zastosowano metodę turniejową, a do krzyżowania procedurę PMX. Jako funkcję przystosowania wprost przyjęto kryterium jakości szeregowania, czyli długość uszeregowania. Zastosowanie podstawowego schematu nie przyniosło zadowalających wyników, dlatego zaproponowano i opracowano nowe wersje zastosowania algorytmu ewolucyjnego. W obu z nich w odmienny sposób, w stosunku do klasycznego schematu algorytmu ewolucyjnego, ustalono podstawowe jego parametry jakim są prawdopodobieństwa p_1 i p_2. W pierwszym przypadku dokonano adaptacji tych prawdopodobieństw w trakcie przebiegu algorytmu ewolucyjnego. Odpowiedni algorytm jest określany jako *algorytm ewolucyjny z adaptacją*. Ponadto do wyznaczania tych prawdopodobieństw zastosowano odpowiedni proces uczenia, a uzyskany algorytm rozwiązania nazwano *algorytmem ewolucyjnym z uczeniem*. Wprowadźmy wpierw następujące oznaczenia:

$\boldsymbol{I}_j$ – j-ta populacja algorytmu ewolucyjnego,

i, $i = 1,2,...,I$ – numer osobnika w populacji o liczności I (przyjmujemy, że wszystkie populacje są tak samo liczne),

$Q_j^{(i)}$ – wartość kryterium (funkcji przystosowania) dla i-tego osobnika w j-tej populacji,

$\underline{Q}_j, \overline{Q}_j, \tilde{Q}_j$ – odpowiednio minimalna, maksymalna i średnia wartość kryterium w j-tej iteracji.

4.1. Algorytm ewolucyjny z adaptacją

W tym przypadku prawdopodobieństwa krzyżowania i mutacji są ustalane na podstawie wartości funkcji przystosowania w populacjach bieżącej i poprzedniej, czyli

$$p_{l,j} = \Phi(Q_j^{(1)}, Q_j^{(2)}, ..., Q_j^{(I)}, Q_{j-1}^{(1)}, Q_{j-1}^{(2)}, ..., Q_{j-1}^{(I)}), \quad l = 1,2, \quad j = 2,3,...,J .$$

W pracy Józefczyk (2001c) podano kilka wersji ustalania tych prawdopodobieństw. Najskuteczniejsza z nich, wykorzystująca propozycję przedstawioną w pracy Srinivas *et al*

(1994), jest następująca

$$p_{1,j} = \begin{cases} \rho_1 \dfrac{Q'_j - \underline{Q}_j}{\tilde{Q}_j - \underline{Q}_j}, & \text{dla } Q'_j \le \tilde{Q}_j, \\ \overline{\rho}_1, & \text{dla } Q'_j > \tilde{Q}_j, \end{cases} \tag{14}$$

gdzie Q'_j jest wartością kryterium dla rodziców $i_1, i_2 \in \boldsymbol{I}_j$, obliczoną według zależności $Q'_j = \min\{Q_j^{(i_1)}, Q_j^{(i_2)}\}$ lub $Q'_j = 0{,}5(Q_j^{(i_1)} + Q_j^{(i_2)})$ oraz $\rho_1, \overline{\rho}_1 \in (0, 1]$ są danymi współczynnikami. Złożona postać wyrażenia (14) wynika z konieczności zachowania warunku $0 \le p_{1,j} \le 1$. Rezultatem ustalenia porządku w populacji jest ciąg osobników. Krzyżowaniu podlegają kolejno dwa sąsiednie osobniki począwszy od pierwszego. Po obliczeniu prawdopodobieństwa krzyżowania według (14), gdzie i_1 oraz i_2 oznaczają dwóch sąsiednich osobników rodzicielskich, jest ono porównywane z losowo wygenerowana liczbą z przedziału [0,1]. Do krzyżowania dochodzi, gdy obliczone prawdopodobieństwo jest większe od wygenerowanej liczby. Wówczas, tradycyjnie, osobniki rodzicielskie są w nowej populacji zastępowane przez osobniki potomne. Prawdopodobieństwa mutacji dla poszczególnych osobników są wyznaczone według wzoru

$$p_{2,j} = \begin{cases} \rho_2 \dfrac{Q_j^{(i)} - \underline{Q}_j}{\tilde{Q}_j - \underline{Q}_j}, & \text{dla } Q_j^{(i)} \le \tilde{Q}_j, \\ \overline{\rho}_2, & \text{dla } Q_j^{(i)} > \tilde{Q}_j, \end{cases}$$

gdzie $\rho_2, \overline{\rho}_2 \in (0, 1]$ są danymi współczynnikami. Warunek przeprowadzenia mutacji osobnika $i \in \boldsymbol{I}_j$ jest podobny jak w przypadku krzyżowania. Oznacza to, że obliczona wartość $p_{2,j}$ jest porównywana z wygenerowaną losowo liczbą z przedziału [0,1] i jeśli jest większa od niej, to dochodzi do mutacji osobnika. Proponowany sposób ustalania obu prawdopodobieństw zapewnia, że do następnej populacji są przenoszone osobniki (rozwiązania) o najlepszych wartościach kryterium. Pozostałe rozwiązania uczestniczą w poszukiwaniu rozwiązań

w nowych obszarach zbioru rozwiązań dopuszczalnych.

4.2. Algorytm ewolucyjny z uczeniem

Innym sposobem wyznaczania obu prawdopodobieństw lub tylko jednego z nich jest zastosowanie uczenia. Punktem wyjścia przy uzasadnianiu celowości stosowania uczenia w rozważanym przypadku jest własność polegająca na tym ,że wartości p_1 i p_2, będące parametrami algorytmu ewolucyjnego, mają z jednej strony wpływ na jakość uzyskiwanego wyniku optymalizacji, ale z drugiej strony zależą od danych problemu optymalizacyjnego i (lub) innych parametrów algorytmu ewolucyjnego. W przypadku szeregowania z ruchomymi realizatorami danymi tymi są przede wszystkim czasy dojazdów realizatorów do stanowisk, ale także czasy wykonywania czynności na stanowiskach. Parametrem algorytmu ewolucyjnego, mającym istotny wpływ na wartości p_1 i p_2, jest wielkość populacji I. Podamy teraz prosty przykład wyznaczania prawdopodobieństw krzyżowania i mutacji w procesie uczenia w wersji bez nauczyciela. W kolejnych cyklach uczenia n, $n = 0,1,...$ wartości p_l, $l = 1,2$ są modyfikowane w sposób rekurencyjny na podstawie wartości kryterium jakości szeregowania, oceniającego kolejne zmiany p_1 i p_2. W każdym cyklu uczenia n jest uruchamiany algorytm ewolucyjny w celu wyznaczania wartości kryterium, ale za każdym razem dla nowych, generowanych losowo i niezależnych od wartości z poprzednich iteracji, danych problemu optymalizacyjnego i (lub) parametrów algorytmu ewolucyjnego. Można zaproponować różne algorytmy uczenia. W rozpatrywanym przypadku sprawdzono dwie ich wersje

$$p_l(n+1) = p_l(n) - \mu_l(n) d_l(n), \tag{15}$$

gdzie $\mu_l(n) = c / n^b$, $c, b > 0$ jest zmiennym współczynnikiem o własnościach

$$\mu_l(n) > 0, \quad \sum_{n=0}^{\infty} \mu_l(n) = \infty, \quad \sum_{n=0}^{\infty} \mu_l^2(n) < \infty$$

oraz

$$p_l(n+1) = p_l(n) + \mu_l(n) d_l(n) \sigma_l(n), \tag{16}$$

gdzie $\mu_l(n) = c_l / n^{b_l}$ dla $c_l, b_l > 0$, $\sigma_l(n) = \mathrm{sgn}[d_l(n-1) - d_l(n)]$ oraz do wyznaczania $d_l(n)$ są wykorzystywane wartości funkcji przystosowania, tzn.

$$d_1(n) = \frac{1}{J} \sum_{j=1}^{J} \frac{\tilde{Q}_j(n)}{\overline{Q}_{\bar{j}}(n)}, \qquad d_2(n) = \frac{1}{J^2} \sum_{j=1}^{J} \frac{\tilde{Q}_{\bar{j}}(n)}{\tilde{Q}_j(n)}$$

oraz

$$\tilde{Q}_j(n) = \frac{1}{I} \sum_{i=1}^{I} Q_j^{(i)}(n), \quad j = 1,2,...,J,$$

$\overline{Q}_{\bar{j}}(n) = \max\{\overline{Q}_1(n), \overline{Q}_2(n), ..., \overline{Q}_m, ..., \overline{Q}_j(n)\}$, gdzie $\overline{Q}_m(n) = \max\limits_{i=1,2,...,I} \{Q_m^{(i)}(n)\}$,

$\underline{Q}_{\bar{j}}(n) = \min\{\underline{Q}_1(n), \underline{Q}_2(n), ..., \underline{Q}_m, \ , \underline{Q}_j(n)\}$, gdzie $\underline{Q}_m(n) = \min\limits_{i=1,2,...,I} \{Q_m^{(i)}(n)\}$.

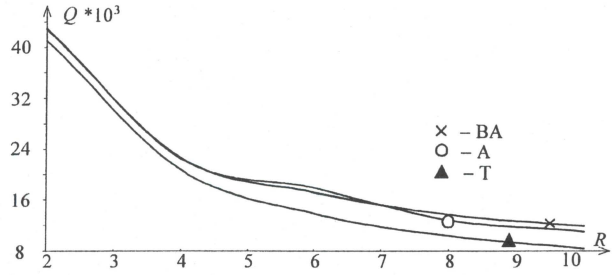
4.3. Badania symulacyjne algorytmów szeregowania

Na rys. 2 przedstawiono przykładowe wyniki badania algorytmu z adaptacją (A) i porównanie z algorytmem ewolucyjnym w podstawowej wersji, tzn. bez adaptacji i bez uczenia (BA), a także z algorytmem tradycyjnym (T) wykorzystującym dekompozycję funkcjonalna, opisaną w rozdz. 3.

Można zauważyć, że zastosowanie adaptacji poprawia wyniki działania algorytmu ewolucyjnego, ale nie jest on lepszy niż algorytm tradycyjny.

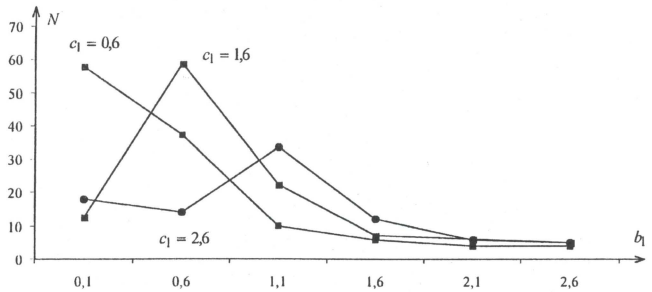
Dla algorytmu ewolucyjnego z uczeniem przeprowadzono dwa rodzaje badań: badanie procesu uczenia oraz badanie efektu uczenia. W pierwszym przypadku określano wpływ parametrów problemu szeregowania, np. liczby zadań H, liczby realizatorów R oraz parametrów algorytmu uczenia, np. współczynników c oraz b na liczbę cykli uczenia N, po której zostanie osiągnięte kryterium stopu

$$\frac{1}{L+1} \sum_{i=0}^{L} \{\zeta^i \sum_{k=1}^{2} [p_k(n-1) - p_k(n)]^2\} < \varepsilon,$$



Rys.2. Porównanie różnych wersji algorytmu ewolucyjnego

gdzie $\zeta \in (0,1]$, L i ε to odpowiednio zadany współczynnik, liczba cykli uczenia uwzględnionych przy obliczaniu dokładności i dokładność uczenia. Przykładowe wyniki przedstawiono na rys. 3. Na wykresie zaprezentowano wyniki symulacji w przypadku, gdy uczeniu podlegało tylko prawdopodobieństwo krzyżowania p_1. Jak można zauważyć na wykresie, przyjęcie zbyt dużych wartości parametrów b_1, c_1 może doprowadzić do zbyt szybkiej zbieżności algorytmu uczenia. Dlatego do dalszych badań przyjęto, że $b_1 = 0{,}6$ i $c_1 = 1{,}6$. Ustalanie prawdopodobieństwa mutacji p_2 w procesie uczenia, dla algorytmów uczenia w formie (15) lub (16) nie przyniosło zadowalających wyników, dlatego w pozostałych badaniach jego wartość przyjmowano za stałą i równą 0,01. Ocenę efektu uczenia przeprowadzono wykorzystując losowo wygenerowany ciąg testujący o długości D. Elementem d, $d = 1,2,...,D$ takiego ciągu była instancja rozważanego problemu z określonymi

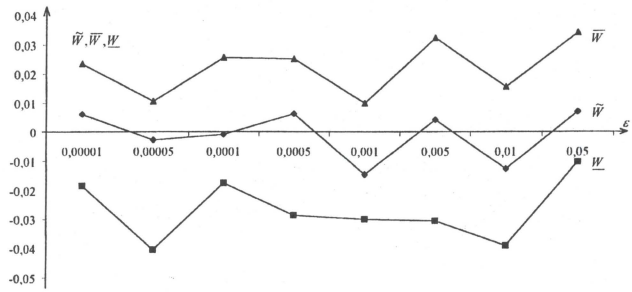


Rys.3. Zależność liczby cykli uczenia N od parametru b_1

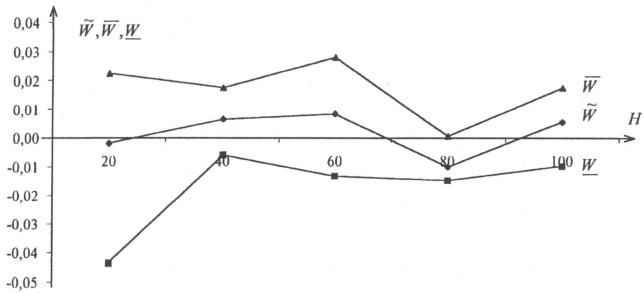
losowo: liczbą zadań, liczbą realizatorów oraz macierzami czasów $\hat{\tau}$ i $\bar{\tau}$. Problem szeregowania dla każdej instancji rozwiązano dwukrotnie, tzn. z wykorzystaniem algorytmów ewolucyjnych z uczeniem oraz bez uczenia. Uzyskane wartości kryteriów jakości $Q(\gamma; p_1')$ oraz $Q(\gamma; p_1'')$ odpowiednio dla algorytmu z uczeniem oraz bez uczenia, gdzie p_1' i p_1'' to odpowiednio wartości prawdopodobieństwa krzyżowania uzyskane w wyniku uczenia oraz przyjęte a priori – zostały wykorzystane do wyznaczenia empirycznych wskaźników jakości oceny algorytmu ewolucyjnego z uczeniem. Zaproponowano trzy następujące postaci takich wskaźników

$$\tilde{W} = \frac{1}{D}\sum_{d=1}^{D} W_d\,, \quad \underline{W} = \min_{d=1,2,\ldots,D}\{W_d\}\,, \quad \overline{W} = \max_{d=1,2,\ldots,D}\{W_d\}\,, \tag{17}$$

gdzie $W_d = (Q(\gamma; p_1') - Q(\gamma; p_1''))/Q(\gamma; p_1'')$. Podobnie jak w przypadku badania procesu uczenia, określono wpływ zmian parametrów problemu szeregowania i algorytmu uczenia na wskaźnik wyrażający cel badania – w tym przypadku na (17). Przykładowe wyniki są przestawione na wykresach na rys. 4 i 5. Zastosowanie uczenia można uznać za skuteczne, gdy



Rys.4. Zależność empirycznych wskaźników jakości od dokładności ε



Rys.5. Zależność empirycznych wskaźników jakości od liczby zadań H

wartości wskaźników w (17) są ujemne. Im mniejsze są to wartości, tym lepiej. Wyniki przedstawione na wykresach wskazują na możliwość kilkuprocentowej poprawy działania algorytmu ewolucyjnego w wyniku uczenia prawdopodobieństw krzyżowania.

5. Uwagi końcowe

W pracy przedstawiono krótki przegląd metod i algorytmów rozwiązania wybranego problemu szeregowania zadań z uwzględnieniem ruchu realizatorów. Skoncentrowano się na prostym problemie szeregowania zadań niezależnych i niepodzielnych na realizatorach dowolnych z kryterium w postaci długości uszeregowania. Dokonanie dekompozycji funkcjonalnej umożliwiło opracowanie algorytmu przybliżonego. Kolejny kierunek badań dotyczył zastosowania do konstrukcji algorytmów metod sztucznej inteligencji. Po przeprowadzeniu analizy możliwości zastosowania różnych konkretnych technik i algorytmów sztucznej inteligencji (np. rozważano możliwość stosowania sztucznych sieci neuronowych) wybrano do dalszych badań algorytmy genetyczne (a ogólniej – algorytmy ewolucyjne). W trakcie badań dotyczących zastosowania podejścia ewolucyjnego do rozwiązania rozpatrywanego zagadnienia szeregowania zauważono istotny wpływ wartości parametrów algorytmu ewolucyjnego na uzyskiwane wyniki optymalizacji. Dlatego opracowano kilka wersji (sposobów) doboru tych

parametrów, przy czym szczególną uwagę zwrócono na prawdopodobieństwa krzyżowania i mutacji.

Sprawa kontynuacji zastosowań algorytmów ewolucyjnych a szerzej algorytmów sztucznej inteligencji do rozwiązywania różnych zagadnień szeregowania zadań z uwzględnieniem ruchu pozostaje otwarta. Bardziej istotne, jak się wydaje, jest jednak rozwijanie innego sposobu wyznaczania decyzji dla rozważanego dyskretnego systemu produkcyjnego. Wiąże się on z inną interpretacją problemu szeregowania z ruchomymi realizatorami. Jak wiadomo, zasadnicza różnica między tym problemem a problemem bez ruchu realizatorów polega na braku apriorycznej znajomości czasów dojazdów realizatorów do stanowisk, a w konsekwencji czasów wykonywania zadań. W dalszych badaniach zakłada się, że czasy wykonywania zadań nie są deterministyczne, co w konsekwencji prowadzi do niedeterministycznych (a ogólniej niepewnych) problemów szeregowania. Mogą być stosowane różne opisy niedeterminizmu. W opisach probabilistycznych odpowiednie czasy są traktowane jako realizacje zmiennych losowych. Rezultaty z tego zakresu przedstawił m.in. Józefczyk (1999). W innym podejściu zakłada się, że czasy dojazdów mogą przyjmować wartości z danych przedziałów lub skończonych zbiorów wartości (Józefczyk (2001b)). Po przyjęciu minimaksowych kryteriów jakości szeregowania uzyskuje się tzw. algorytmy szeregowania odpornego. Zastosowanie opisów w postaci liczb rozmytych oraz tzw. zmiennych niepewnych (Bubnicki (2001)) jest w trakcie opracowywania. Warto zauważyć, że przyjęcie niedeterministycznych opisów czasów wykonania zadań jest w tym przypadku szczególnie uzasadnione, co wynika z interpretacji problemu szeregowania z uwzględnieniem ruchu. Interpretację tę w sposób szczegółowy przedstawił Józefczyk (2001c). Ponadto, w przypadku niektórych opisów niedeterministycznych ich przyjęcie umożliwia uzyskanie odpowiedniego klasycznego niedeterministycznego zagadnienia szeregowania, zamiast deterministycznego zagadnienia szeregowania z uwzględnieniem ruchu.

Literatura

Baeck, T. (1996) *Evolutionary Algorithms in Theory and Practice*. Oxford Univ. Press, NY.

Banaszak, Z., Józefczyk, J. (2001) Evolutionary algorithm with learning for scheduling of manufacturing tasks with moving executors. w: *XIV International Conference on Systems Science* (red. Z. Bubnicki, A. Grzech), Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, **II**, 187-194.

Browne, S., Yechiali, U. (1990) Scheduling deteriorating jobs on a single processor. *Operations Research*, **38**, 495-497.

Bubnicki, Z. (2001) Uncertain variables and their applications for a class of uncertain systems. *International Journal of Systems Science*, **32**, 651-659.

Chen, Z.L. (1996) Paralell machine scheduling with time dependent processing times. *Discrete Applied Mathematics*, **70**, 81-93.

Foncesca, C.M. (1995) An overview of evolutionary algorithms in multiobjective optimisation. *Evolutionary Computation*, **3**, 1–16.

Janiak, A. (1999*) Wybrane Problemy i Algorytmy Szeregowania Zadań i Rozdziału Zasobów*. Akademicka Oficyna Wydawnicza PLJ, Warszawa.

Józefczyk, J. (1996) *Szeregowanie Zadań w Kompleksie Operacji z Uwzględnieniem Ruchu Realizatorów*. Oficyna Wydawnicza Polit. Wrocławskiej, Wrocław.

Józefczyk, J. (1997) An algorithm for scheduling tasks on moving executors in complex operation systems. *Proc. of 1st IFAC Workshop on Manufacturing Systems MIM'97*, Wiedeń, Austria, 139-144.

Józefczyk, J. (1999) On the problem of scheduling tasks on moving executors with random processing times. *Systems Science*, **25**, 5–13.

Józefczyk, J. (2000) Zastosowanie algorytmu genetycznego w problemie szeregowania zadań z ruchomymi realizatorami. w: *Inżynieria Wiedzy i Systemy Ekspertowe* (red.: Z. Bub-

nicki, A. Grzech), Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 29–36.

Józefczyk, J. (2001a) Algorithm for scheduling of tasks on moving executors with uncertain processing times. *Proc. of 6th European Control Conference ECC '2001*, Porto, Portugalia, 1034-1039.

Józefczyk, J. (2001b) Scheduling tasks on moving executors to minimise the maximum lateness. *European Journal of Operational Research*, **131**, 171–187.

Józefczyk, J. (2001c) *Wybrane Problemy Podejmowania Decyzji w Kompleksach Operacji*, seria „Monografie Komitetu Automatyki i Robotyki PAN", tom 2., Oficyna Wydawnicza Politechniki Wrocławskiej, Warszawa-Wrocław.

Medskev, L.R. (1995) *Hybrid Intelligent Systems*. Kluwer Academic Publishers, Boston.

Słowiński, R., Hapke, M. (red.) (2000) *Scheduling under Fuziness*. Physica-Verlag.

Srinivas, M., Patnaik, M. (1994) Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Trans. on SMC*, **24**, 656-666.