AUTOMATYKA • 2009 • Tom 13 • Zeszyt 2

Mariusz Makuchowski*, Adam Tyński*

Automatyczna mutacja w algorytmach ewolucyjnych

1. Wprowadzenie

Ogólna idea algorytmów genetycznych opisana w [1, 2] jest bardzo klarowna i owocuje łatwymi w implementacji algorytmami ewolucyjnymi. Największym problemem w tego rodzaju algorytmach jest właściwy dobór parametrów sterujących. Konstruktorzy dedykowanych algorytmów ewolucyjnych najczęściej dobierają wartości parametrów w sposób doświadczalny. Wadą takiego podejścia jest nie tylko pracochłonność, ale i fakt strojenia algorytmu tylko pod konkretny zestaw instancji. By zmniejszyć wagę wymienionej wady, w pracy przedstawia się ogólną ideę automatycznego doboru jednego z istotniejszych parametrów algorytmów genetycznych – prawdopodobieństwa mutacji. Prezentowane podejście jest uniwersalną ideą, mogącą mieć zastosowanie w wszelkiego rodzaju algorytmach ewolucyjnych.

2. Opis problemu

Testowane algorytmy dedykowane są dla problemu gniazdowego (*jobshop problem*) z dodatkowym ograniczeniem bez czekania (*no-wait*). Jako kryterium optymalizacji przyjmuje się moment zakończenia wykonywania wszystkich zadań (*makespan*). Rozważany problem w trójpolowej notacji Grahama [3] oznaczany jest przez $J|\ no\text{-}wait\ |C_{\max}$. Problem ten różni się od swego klasycznego odpowiednika (przez problem klasyczny rozumie się problem gniazdowy bez dodatkowych ograniczeń) wymogiem rozpoczęcia wykonywania operacji dokładnie w chwili zakończenia wykonywania się poprzednika technologicznego. Ograniczenie te najczęściej występują w gałęziach przemysłu, w których przerabiany produkt zmienia szybko swoje własności fizyczno-chemiczne, np. produkcja leków [4], żywności [5], wytop stali [6] czy wyrób elementów betonowych [7], choć także spotykane są w innych dziedzinach, np. testowanie półprzewodników [8] czy systemach komputerowych [9].

* Instytut Informatyki, Automatyki i Robotyki, Politechnika Wrocławska

W celu zrozumienia dalszej część pracy, nie jest potrzebna precyzyjna postać matematycznego modelu problemu, dlatego nie zostanie on tu przytoczony. Prezentowana metoda, modyfikacji algorytmu ewolucyjnego, nie jest w żaden sposób związana z konkretnym problemem, lecz jest ogólną ideą wprowadzania zaburzeń w pokoleniu algorytmu ewolucyjnego. Wybrany problem gniazdowy z ograniczeniem bez czekania jest tylko przykładowym zagadnieniem optymalizacji dyskretnej i służy nam do przetestowania proponowanej idei. Ważnym elementem tego problemu jest tylko sposób kodowania rozwiązania, gdyż ma on bezpośredni wpływ na postać operatora mutacji. Rozwiązaniem analizowanego problemu jest permutacja zbioru zadań do wykonania. Permutacja ta wykorzystywana jest następnie przez pewien algorytm konstrukcyjny [10] budujący ostatecznie właściwe rozwiązanie. Ponieważ rozwiązanie reprezentowane jest w postaci kolejności zadań, w pracy analizowane są cztery typowe operatory mutacji działające na permutacjach.

3. Mutacja w algorytmach ewolucyjnych

Pomimo iż mutacja w algorytmach genetycznych zachodzi sporadycznie, ma ona ogromne znaczenie dla efektywności całego algorytmu. Podstawowe cele mutacji to:

- tworzenie osobników różniących się częściowo, ale w istotny sposób od rodziców, co umożliwia przeniesienie poszukiwań w coraz to nowe części przestrzeni rozwiązań;
- utworzenie osobnika o cesze niewystępującej w pokoleniu, co umożliwia znalezienie rozwiązania lokalnie optymalnego nawet w przypadku zatracenia się w pokoleniu pewnej optymalnej cechy tego rozwiązania;
- przeciwdziałanie stagnacji obliczeń, to znaczy sytuacji, w której całe pokolenie zdominowane jest przez jeden rodzaj mało różniących się między sobą osobników.

Zanim przejdziemy do omówienia metody automatycznego strojenia algorytmu genetycznego wprowadzimy najpierw pojęcie poziomu mutacji. Byt ten jest bezpośrednio powiązany z dobrze znanym z literatury prawdopodobieństwem mutacji. Różnice pomiędzy tymi bytami mogą wydawać się subtelne, jednakże są bardzo istotne ze względu na przeprowadzane w dalszej części pracy badania. Poziom mutacji L określa z definicji względną liczbę mutacji w całym pokoleniu (*liczba wszystkich mutacji* = L·*liczba wszystkich genów*), natomiast prawdopodobieństwo mutacji P z definicji jest prawdopodobieństwem tego, że pojedynczy gen zostanie zmutowany. Zauważmy teraz, że w zależności od sposobu rozwiązania i stosowanego operatora mutacji, mutowany gen może zmieniać swoją wartość w bardzo ograniczony sposób, podczas gdy w losowym rozwiązaniu gen ten może przyjmować wartości ze znacznie liczniejszego zbioru. W takim przypadku nawet dla $P = 1$ (każdy gen zostaje zmutowany) może istnieć duże podobieństwo pomiędzy genotypem pierwotnym a zmutowanym. Przykładem powyższego zjawiska jest sytuacja, w której cały genotyp składa się z jednego chromosomu będącego permutacją. Jeżeli w takim przypadku mutacja

polega na zamianie w permutacji miejscami dwóch sąsiednich elementów, wówczas nawet podczas mutacji każdego genu istnieje duże podobieństwo permutacji pierwotnej x i permutacji zmutowanej y. Dowodem tego jest odległość pomiędzy tymi permutacjami $d(x, y)$, rozumiana jako najmniejsza liczba zamian sąsiednich potrzebna do przeprowadzenia permutacji x w permutacje y. Z definicji odległości oraz ilości wykonanych mutacji równej n ilości elementów w permutacji, maksymalna odległość pomiędzy x i y może wynosić co najwyżej n; $d(x, y) \leq n$. Odległość n jest stosunkowo mała w porównaniu z średnią odległością pomiędzy losową permutacją z a permutacją x wynoszącą $AV\,E(d(x, z)) = n\cdot(n - 1)/4$, przy czym największa możliwa odległość pomiędzy permutacjami jest dwa razy większa. Oznacza to, że permutacja x i otrzymana poprzez mutacje każdego genu permutacja y są do siebie (w sensie odległości d) podobne. Z powyższej własności wynika bezpośrednio, że mutowanie zmutowanych już wcześniej genów powoduje wprowadzanie jeszcze większych zakłóceń w genotypie osobnika. Fakt ten w żaden sposób nie jest uwzględniany przez parametr P, który informuje o prawdopodobieństwie przynajmniej jednokrotnej mutacji genu. Przeciwnie poziom mutacji o wartości większej niż 1 świadczy, że statystycznie w jednym osobniku dokonuje się więcej mutacji niż posiada on genów. W praktyce jednak optymalny poziom mutacji L jest na poziomie do kilku procent i w przybliżeniu jest niemal dokładnie równy P, a rozważania dla wartości L zbliżających się do 1 i większych mają charakter tylko teoretyczny. Przy założeniu, iż każda mutacja zmienia dokładnie k genów, można napisać zależność prawdopodobieństwa mutacji P od g ilości wszystkich genów w pokoleniu i L poziomu mutacji:

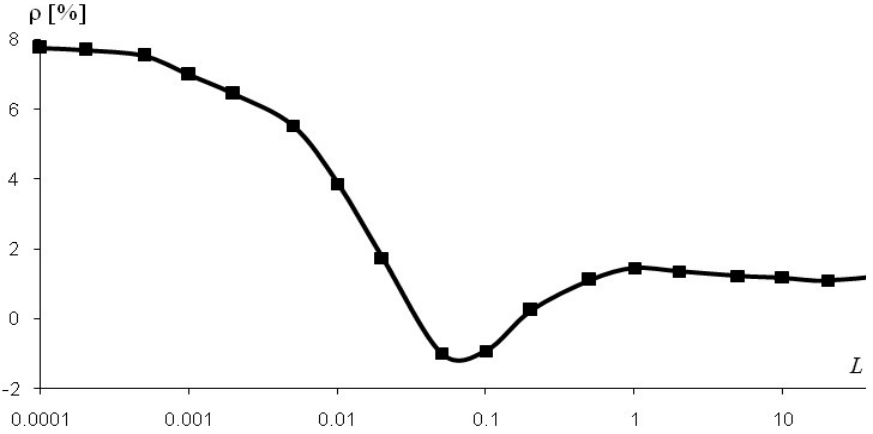
$$P(g,k,L) = 1 - \left(\frac{g-k}{g}\right)^{g\cdot L} \tag{1}$$

W pracy rozważa się cztery rodzaje operatorów mutacji:

1) *small swap*: polegający na zamianie losowych dwóch sąsiednich elementów permutacji, oznaczany dalej w skrócie przez *SSw*;
2) *swap*: polegający na zamianie miejscami dwóch różnych losowych elementów permutacji, oznaczany dalej w skrócie prze *Swp*;
3) *insert*: polegający na wyciągnięciu losowego elementu z permutacji i ponownym włożeniu go na losową pozycję (z wykluczeniem pierwotnej pozycji), oznaczany dalej w skrócie przez *Ins*;
4) *invert*: polegający na odwróceniu kolejności elementów losowej części permutacji, oznaczany dalej w skrócie przez *Inv*.

3.1. Mutacja – podejście klasyczne

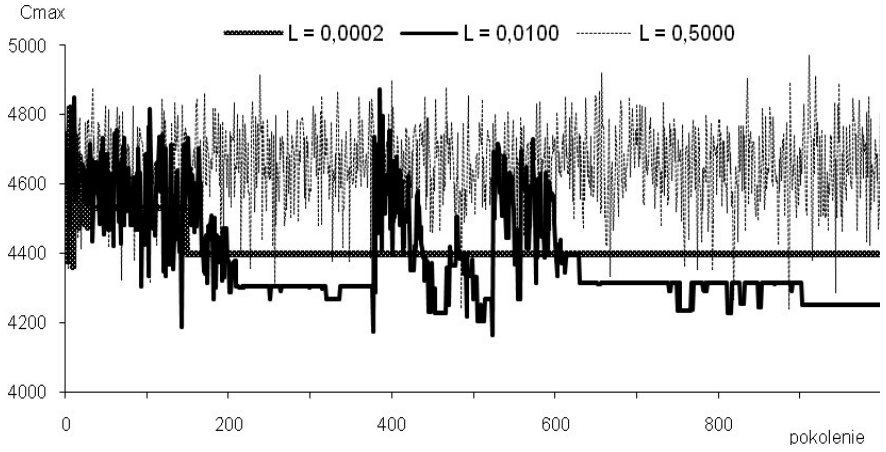
Z badań przeprowadzonych przez autorów wynika, iż wartość poziomu mutacji ma decydujący wpływ na efektywność algorytmu. Przykładowy średni błąd uzyskiwanych rozwiązań względem rozwiązań referencyjnych w zależności od wartości poziomu mutacji przedstawiony jest na rysunku 1.



Rys. 1. Wpływ poziomu mutacji na średni błąd algorytmu ewolucyjnego względem rozwiązań referencyjnych

Optymalna wartość poziomu mutacji zależy zarówno od stosowanego operatora mutacji, jak i od danych instancji problemu. W celu dokładniejszego zrozumienia zjawisk zachodzących w algorytmie genetycznym, na rysunku 2 przedstawiono przebieg algorytmu ewolucyjnego o zbyt małym, dobrym i zbyt dużym poziomie mutacji. Wartość $c_{\max}$ pokolenia rozumiana jest jako najmniejsza wartość funkcji celu wyliczonych dla wszystkich osobników badanej populacji bieżącego pokolenia. Przy zbyt małej wartości poziomu mutacji w algorytmie GA następuje stagnacja obliczeń (pierwszy przebieg na rys. 2). Dzieje się tak dlatego, ponieważ pokolenie zdominowane jest przez tak zwane superosobniki, dominujące zarówno jakościowo jak i ilościowo. Osobniki te są praktycznie identyczne, co najwyżej różnią się między sobą w sposób nieistotny dla rozwiązania i kryterium funkcji celu. Superosobniki są na tyle dobre w sensie wartości funkcji przystosowania, iż często dochodzi do krzyżowania się ich między sobą, tworzy się nowy superosobnik w następnym pokoleniu oraz wszystkie pozostałe krzyżowania w pokoleniu generują potomstwo dużo słabsze w sensie wartości funkcji przystosowania. W konsekwencji po selekcji naturalnej nowe pokolenie zdominowane jest znowu przez superosobniki identyczne jak w pokoleniu wcześniejszym. Nastąpiła więc stagnacja obliczeń, algorytm popadł w pewien rodzaj optimum lokalnego z bardzo małymi szansami na jego opuszczenie. Przy zbyt wysokiej wartości poziomu mutacji, algorytm GA wykazuje cechy algorytmu sprawdzającego losowe rozwiązania (trzeci przebieg na rys. 2). Jest tak dlatego, iż w nowo powstającym osobniku zachodzi stosunkowo duża liczba mutacji, co skutkuje tym, że zatracają się wszystkie cechy przeniesione z jego rodziców. Przestaje więc działać mechanizm dziedziczenia, a w jego miejscu pojawia się generowanie losowych osobników. W tej sytuacji nowo powstające pokolenie nie jest coraz lepszą odmianą osobników z wcześniejszych pokoleń, lecz zbiorem losowych rozwiązań mających niewiele wspólnego z wcześniejszymi pokoleniami. Sytuacje w której poziom mutacji jest optymalny, czyli na tyle duży by nie zaszło zjawisko stagnacji obliczeń

oraz na tyle mały by nowo powstające osobniki dziedziczyły cechy swoich rodziców, przedstawiony jest w drugim przebiegu rysunku 2. Na wykresie tym widać podstawowe zjawiska zachodzące w poprawnie wysterowanym algorytmie genetycznym. Są nimi szybkie „schodzenie" algorytmu do minimum lokalnego oraz efektywna dywersyfikacja obliczeń uwidaczniająca się opuszczaniem znalezionych minimów lokalnych. Ponieważ dokładne wyznaczenie optymalnej wartości poziomu mutacji w sposób analityczny na dzisiejszym poziomie wiedzy jest niemożliwe, jedyna metoda jego wyznaczenia to eksperymenty numeryczne.



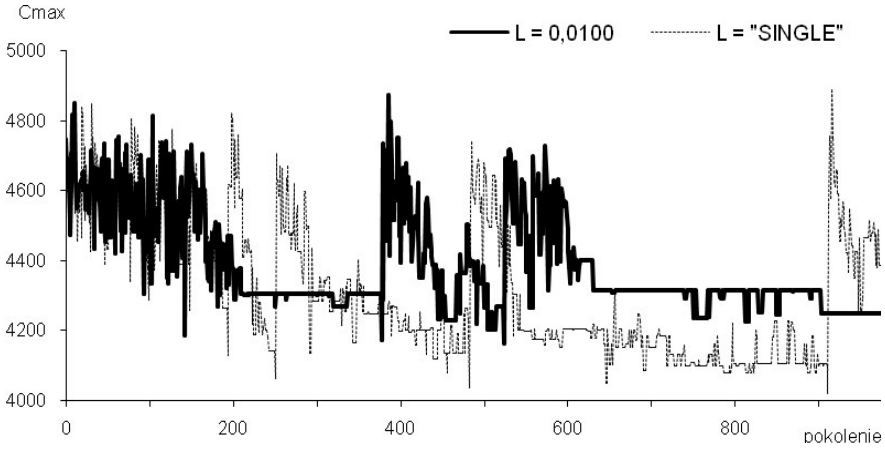
Rys. 2. Wpływ poziomu mutacji na przebieg poszukiwań w algorytmie ewolucyjnym

3.2. Mutacja – automatyczna

W niniejszej pracy proponowany sposób mutacji odbiega od jego klasycznego pierwowzoru. Dokładniej, zamiast dokonywać mutacji genów z ustalonym (na stałe lub dynamicznie zmienianym) prawdopodobieństwem, proponuje się wstępną analizę populacji każdego pokolenia, a następnie na jej podstawie wyznaczenie osobników, które należy poddać mutacji. Metoda ta nie tylko dobiera automatycznie ilość mutacji w pokoleniu, ale także wskazuje osobniki, które należy zmodyfikować. Proponowana strategia bazuje na dwóch spostrzeżeniach:

1) W przypadku, gdy cała populacja jest mocno zróżnicowana, nie ma potrzeby mutowania jej osobników. Mutacja w takim przypadku pozbawia ich tylko odziedziczonych cech.
2) W przypadku pojawienia się w pokoleniu kilku superosobników, należy dokonać ich mutacji. Taka mutacja zapewnia uniknięcia stagnacji obliczeń poprzez eliminacje identycznych osobników. Ponadto umożliwia szybkie dojście do optimum lokalnego (w przypadku gdy superosobnicy reprezentują rozwiązanie bliskie rozwiązaniu lokalnie optymalnemu). Ponadto umożliwia łatwe opuszczenie minimum lokalnego, ponieważ w pokoleniu pozostaje tylko jeden osobnik reprezentujący takie rozwiązanie.

Strategia postępowania wydaje się oczywista. Należy zidentyfikować wszystkie klony, a następnie dokonać ich mutacji. Klonem nazywamy osobnika podobnego w pewnym sensie do innego osobnika istniejącego w danym pokoleniu. Dla takiego podejścia, choć intuicyjnego, nie udało nam się znaleźć analogii w świecie przyrody. Przy implementacji proponowanej strategii do rozstrzygnięcia pozostaje między innymi sposób detekcji klonów. Do detekcji klonów zastosowano najprostszą metodę – to znaczy poprzez porównywanie wartości funkcji przystosowania osobników. Osobnicy o tej samej wartości funkcji przystosowania traktowani są jak klony. Klony wybrane do mutacji powinny zostać zmutowane efektywnie, to znaczy w taki sposób, by mutacja rzeczywiście zmieniła nie tylko genotyp, ale także fenotyp osobnika. W tym celu można przykładowo kontrolować zmianę wartości funkcji przystosowania mutowanych osobników. Efektami ubocznym wynikającymi z wykrywania klonów oraz testowaniem zmiany mutowanego osobnika na podstawie wartości funkcji celu odpowiadających im rozwiązań jest możliwość stwierdzenia identyczności osobników mimo różnych fenotypów oraz dodatkowe bardzo czasochłonne obliczenia wyznaczające wartość funkcji celu. Pierwsze z wymienionych zjawisk zachodzi stosunkowo rzadko i nie powoduje żadnych poważnych zaburzeń w algorytmie, natomiast drugie zdecydowanie spowalnia jego prace. Zaproponowana wcześniej strategia mutacji osobnika (polegająca na jego mutacji aż do zmiany wartości funkcji celu) jest bardzo wolna i zostaje nazwana strategią *FULL*. Przyspieszenie pracy algorytmu można dokonać poprzez redukcję strategii *FULL* do strategii nazwanej *SINGLE*. Polegającej ona na jednokrotnej mutacji i jednokrotnym obliczeniu zmodyfikowanej wartości funkcji przystosowania (bez względu na to czy nastąpiła zmiana jej wartości czy też nie). Kolejna strategia praktycznie niespowalniająca pracę algorytmu jest strategią *BLIND* i polega ona na jednokrotnej mutacji danego osobnika bez ponownego wyliczenia wartości funkcji przystosowania. Tak zmutowany osobnik podlega dalszej selekcji według pierwotnej wartości funkcji.



Rys. 3. Przebieg algorytmu ewolucyjnego z automatyczną mutacją typu „SINGLE”

Przykładowy przebieg algorytmu ewolucyjnego z automatyczną mutacją typu *SINGLE* został przedstawiony na rysunku 3. Z analizy tego wykresu można wnioskować, iż algorytm znajduje minima lokalne, dość dokładnie przeszukuje ich otoczenie a po pewnym czasie opuszcza badany region przenosząc poszukiwania w inną część przestrzeni rozwiązań.

4. Badania numeryczne

W tym punkcie zostaną numerycznie zweryfikowane opisywane w pracy własności. Badania ograniczone są tylko do jednego problemu optymalizacji dyskretnej – problemu gniazdowego z ograniczeniem bez czekania. Autorzy zdecydowali się na taki wybór, ponieważ problem ten jest uważany przez badaczy za jeden z trudniejszych problemów optymalizacji dyskretnej.

4.1. Przykłady testowe

Wszystkie eksperymenty numeryczne przedstawione w tej pracy przeprowadzone zostały na 40 literaturowych przykładach testowych la01–la40. Przykłady te dedykowane są klasycznemu problemowi gniazdowemu i uważane są przez badaczy za wyjątkowo trudne. Oczywiście w literaturze występuje także wiele innych zestawów trudnych przykładów testowych, jednakże zestaw zaproponowany przez Lawrence'a wydaje się wystarczająco duży i zróżnicowany. Jest on podzielony na osiem grupy po pięć instancji każda. W każdej grupie wszystkie przykłady charakteryzują się jednakowym rozmiarem, a dokładniej jednakową liczbą zadań n, maszyn m i operacji o. Ponadto cechą charakterystyczną tych przykładów jest to, że liczba wszystkich operacji wynosi dokładnie $o = n \cdot m$ oraz każde zadanie składa się dokładnie z m operacji wykonywanych na różnych maszynach. W dalszej części rozdziału poszczególne grupy oznaczane są poprzez podanie rozmiaru instancji w nich zawartych $n \times m$. Prezentowane w tabelach wyniki odnoszące się do całych grup są wartościami średnimi odpowiednich wartości otrzymanych dla wszystkich przykładów z danej grupy.

4.2. Badania numeryczne

Przeprowadzone testy polegają na porównaniu średniej wartości ρ błędu generowanych przez algorytm genetyczny wysterowany przez dobrany poziom mutacji L^* oraz automatyczną mutację ze strategią *SINGLE*, *FULL* i *BLIND*. Dobierany eksperymentalnie poziom mutacji L^* wybierany jest ze zbioru $L^* \in$ {0,0001, 0,0002, 0,0005, 0,001, 0,002, 0,005, 0,01, 0,02, 0,05, 0,1, 0,2, 0,5, 1, 2, 5, 10, 20, 50, 100} jako poziom dla którego pojedynczy przebieg algorytmu okazał się najbardziej korzystny. Ze względu, iż wartość poziomu L^* wyznaczona jest eksperymentalnie jest ona tylko pewnym przybliżeniem wartości optymalnego poziomu. Omawiane badania zostały wykonane dla każdego z testowanych operatorów mutacji a wszystkie otrzymane wyniki zamieszczone zostały w tabelach 1 i 2. Kolejno, tabela 1 zawiera wartość optymalnego poziomu L^* i średnią jakość dla algorytmu o mutacji

na poziomie L^* natomiast tabela 2 zawiera średnie wartości algorytmu z automatyczną mutacją typu *FULL*, *SINGLE* i *BLIND*. Wartość ρ dla danego rozwiązania x obliczana jest ze wzoru (1) względem rozwiązania x^* otrzymanego literaturowym [11] algorytmem *GASA*.

$$\rho(x) = \frac{c_{\max}(x) - c_{\max}(x^*)}{c_{\max}(x^*)} \cdot 100\% \tag{2}$$

Tabela 1
Wyznaczony poziom mutacji L^* oraz odpowiadająca mu średnia jakość generowanych rozwiązań ρ [%], testowanych operatorów mutacji

Grupa	Optymalny poziom mutacji L^*				Średni błąd ρ dla mutacji L^*			
n × m	SSw	Swp	Ins	Inv	SSw	Swp	Ins	Inv
10 × 5	2.00	0.20	0.10	0.10	-1.91	-2.63	-1.56	-1.47
15 × 5	0.50	0.10	0.10	0.10	0.31	1.94	1.29	1.77
20 × 5	1.00	0.05	0.05	0.05	-0.28	-2.01	-0.16	-1.22
10 × 10	1.00	0.50	0.20	1.00	-5.40	-5.29	-5.29	-5.47
15 × 10	0.50	0.10	0.10	0.10	-1.39	0.09	-0.73	0.78
20 × 10	0.50	0.05	0.10	0.05	-1.22	-2.69	-2.10	-2.15
30 × 10	0.20	0.05	0.05	0.05	-5.68	-4.69	-6.54	-3.88
15 × 15	1.00	0.05	0.10	0.10	-1.74	-1.86	-2.06	-0.56
wszystkie	0.50	0.05	0.10	0.05	-1.72	-0.83	-1.27	-0.86
średnia	0.84	0.14	0.10	0.19	-2.16	-2.14	-2.14	-1.53

Tabela 2
Średnia jakość ρ [%] generowanych rozwiązań dla badanych metod automatycznej mutacji

Grupa	mutacja *FULL*				mutacja *SINGLE*				mutacja *BLIND*			
n × m	SSw	Swp	Ins	Inv	SSw	Swp	Ins	Inv	SSw	Swp	Ins	Inv
10 × 5	2.25	-1.79	-1.84	-1.68	1.61	-1.77	-2.47	-1.99	6.12	-2.04	-1.96	-1.65
15 × 5	6.45	1.16	1.29	2.87	5.64	0.69	1.46	2.72	6.14	0.64	1.06	0.32
20 × 5	2.61	-1.77	-1.80	-0.71	2.77	-0.68	-1.68	-1.82	5.13	-1.80	-1.67	-0.95
10 × 10	-0.25	-3.08	-2.39	-3.94	-0.90	-3.93	-3.53	-3.89	-0.77	-2.71	-3.44	-3.39
15 × 10	5.51	0.46	-0.74	-0.79	5.91	-0.34	2.16	0.69	5.63	0.62	1.04	0.33
20 × 10	1.90	-2.11	-1.99	-1.08	5.28	-0.56	-0.90	-1.58	3.94	-1.33	-0.93	-0.56
30 × 10	-2.96	-6.47	-6.19	-4.90	-1.03	-6.15	-7.55	-5.33	0.17	-6.79	-5.05	-5.80
15 × 15	4.82	-0.45	-2.35	-2.19	5.09	-1.93	-0.73	-0.15	7.05	-0.77	0.67	-0.97
wszystkie	2.54	-1.76	-2.00	-1.55	3.05	-1.83	-1.66	-1.42	4.18	-1.77	-1.29	-1.58

Ponadto komentarza wymagają dwa ostatnie wiersze *wszystkie* i *średnia* zawarte w tabeli 1. Pierwszy z nich zawiera wartości parametrów uzyskane dla najlepiej dobranego,

jednakowego dla wszystkich grup, poziomu mutacji. Ostatni wiersz zawiera natomiast średnie wartości parametrów uzyskane w przypadku, gdy dla każdej z grup przykładów poziom mutacji wyznaczany był indywidualnie.

4.3. Ocena wyników

Na początku pracy wypunktowane zostały główne cele stosowania mutacji w algorytmach genetycznych. Z obserwacji przykładowego przebiegu algorytmu z automatycznie dobieraną mutacją (rys. 3) widać dokładnie, iż w prezentowanym przykładzie spełnia ona wszystkie wymagania dotyczące prawidłowej mutacji. Główną wadą zaproponowanego podejścia okazał się fakt, iż istnieją operatory mutacji (w analizowanym przypadku operator *SSw*) dające mierne rezultaty prezentowanego automatycznego doboru mutacji. Niemniej dla pozostałych trzech operatorów *Swp*, *Ins* i *Inv*, wyniki zaproponowanej automatycznej mutacji są bardzo dobre. Z analizy wyżej wymienionych trzech operatorów mutacji wynika, iż w rozważanym problemie, każdy z wariantów automatycznej mutacji jest statystycznie lepszy niż najlepiej dobrana mutacja klasyczna. Ponadto, w przypadku gdy najlepszy możliwy poziom mutacji dobierany jest indywidualnie dla każdej z grup przykładów, algorytmy z automatyczną mutacją wypadają tylko nieznacznie słabiej. Należy tu zauważyć, że ostatnie z porównań stawia proponowany algorytm w niekorzystnej pozycji. Jest tak dlatego, gdyż omawiany test polegał na jednokrotnym uruchomieniu wszystkich algorytmów i otrzymane wyniki są częściowo losowe. W algorytmie o stałym poziomie mutacji zdarza się, że dla jakiegoś poziomu mutacji L' wartości ρ z pojedynczych przebiegów będą korzystniejsze niż dla poziomu L^*. W takim przypadku do oceny porównawczej uwzględnia się najlepsze z otrzymanych rezultatów. Eksperyment ten eliminuje więc pojedyncze „wpadki" algorytmu klasycznego oraz wybiera wyjątkowo „udane przypadki" przebiegów o innych poziomach mutacji. Przeciwnie w algorytmie z automatyczną mutacją do oceny podawany jest zawsze tylko jeden przebieg, w którym „wpadki" zawsze zmniejszają jakość badanego współczynnika ρ.

5. Podsumowanie

Prezentowaną w pracy metodę automatycznego strojenia można stosować dla dowolnego algorytmu ewolucyjnego. Opiera się ona na bytach występujących w każdym algorytmie genetycznym niezależnie od rozwiązywanego problemu. Automatyczny dobór prawdopodobieństwa mutacji ze wskazywaniem osobników do mutacji odbywa się on-line, to znaczy na bieżąco podczas pracy algorytmu. Jakość proponowanego automatycznego strojenia algorytmu zweryfikowana została na przykładzie algorytmu genetycznego dedykowanego problemu gniazdowego z dodatkowym ograniczeniem bez czekania. Wyniki numeryczne wykazują, iż w badanym problemie efektywność algorytmu z samostrojeniem jest na poziomie algorytmu sterowanego klasycznie z możliwie najlepiej dobranymi parametrami sterującymi. Co więcej, prezentowany algorytm jest znacznie efektywniejszy niż ewolucyjne algorytmy literaturowe [11].

Literatura

[1] Holland J.H., *Genetic Algorithms*. Scientific American, 267, 1992, 44.
[2] Goldberg D.E., *Algorytmy genetyczne i ich zastosowania*. WNT, Warszawa 1995.
[3] Graham R., Lawler E., Lenstra J., Rinnooy Kan A., *Optimization and approximation in deterministic sequencing and scheduling: a survey*. Annals of Discrete Mathematics, 5, 1979, 287.
[4] Raaymakers W., Hoogeveen J., *Scheduling multipurpouse batch process industries with no-wait restrictions by simulated annealing*. European Journal of Operational Research, 126, 2000, 131.
[5] Hall N., Sriskandarajah C., *A survey of machine scheduling-problems with blocking and no-wait in process*. Operations Research, 44 (3), 1996, 510.
[6] Wismer D.A., *Solution of the flowshop scheduling-problem with no intermediate queues*. Operations Research, 20, 1972, 689.
[7] Grabowski J., Pempera J., *Sequencing of jobs in some production system*. European Journal of Operational Research, 125, 2000, 535.
[8] Ovacik I., Uzsoy R., *Decompisition methods for complex factory scheduling problems*. Operations Research Letters, 31, 2003, 308–318.
[9] Reddi S., Ramamoorth C., *A scheduling problem*. Operational Research Quart., 24, 1973, 441–446.
[10] Bożejko W., Makuchowski M., *A fast hybrid tabu search algorithm for the no-wait job shop problem*. Computers & Industrial Engineering (2008), doi:10.1016/j.cie.2008.09.023.
[11] Schuster C., Framinan J., *Approximative procedures for no-wait job shop scheduling*. Operations Research Letters, 31, 2003, 308.