ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1998
Seria: AUTOMATYKA z. 125 Nr kol. 1391

Wojciech CHMIEL
Akademia Górniczo-Hutnicza, Kraków

ALGORYTM EWOLUCYJNY Z OGRANICZONYM WYBOREM OPERATORÓW GENETYCZNYCH*

Streszczenie. W pracy przedstawiono wyniki badań eksperymentalnych procesu optymalizacji realizowanego za pomocą algorytmu genetycznego dla testowego zagadnienia przydziału z kwadratową funkcją celu. Badany algorytm genetyczny realizuje oryginalny proces genetycznego poszukiwania.

EVOLUTIONARY ALGORITHM WITH RESTRICTED CHOICE OF GENETIC OPERATORS

Summary. This paper presents results of experimental examination of optimization process realized with the aid of genetic algorithm on the test example of quadratic assignment problem. The investigated genetic algorithm realizes original genetic search process.

1. Wstęp

Zagadnienia, dla których rozwiązania można sformułować w postaci permutacji, stanowią ważną klasę problemów decyzyjnych w dziedzinie optymalizacji kombinatorycznej.

Klasycznymi typami problemów permutacyjnych są: kwadratowy problem przypisania, kolorowania grafu, zagadnienia harmonogramowania, zagadnienia komiwojażera oraz wiele innych ważnych zagadnień optymalizacyjnych.

Badania nad algorytmami przybliżonymi, dostarczającymi rozwiązań dla zagadnień, w których zastosowanie metod dokładnych jest niemożliwe ze względu na liczność przestrzeni rozwiązań, stanową obecnie jedną z najciekawszych i najszybciej rozwijających się gałęzi nauki.

Artykuł prezentuje rezultaty poszukiwań przybliżonych algorytmów realizujących ewolucyjny proces poszukiwania dla problemów o kwadratowym wskaźniku jakości, zwykle

* Praca jest finansowana przez Komitet Badań Naukowych, grant 11.127.227 (AGH, Kraków).

oznaczanym symbolem *QAP (ang. Quadratic Assigment Problem)*, należących do klasy zagadnień *NP-trudnych*. Algorytm został przetestowany w oparciu o bogaty zbiór przykładowych zadań zaczerpniętych z biblioteki *QAPLIB-A*, będących problemami testowymi przeznaczonymi dla zagadnień aproksymacyjnych.

Należy pamiętać, że *QAP* jest szczególnie trudnym zagadnieniem permutacyjnym. Rozwiązanie tego zagadnienia metodami dokładnymi dla rozmiaru $n > 15$ napotyka na znaczne trudności obliczeniowe, co jest powodem licznych prac w zakresie rozwoju metod przybliżonych dla tego zagadnienia (por. [1], [3], [4], [5]).

Zagadnienie *QAP* sformalizujemy w następujący sposób. Dany jest zbiór $N=\{1,...,n\}$ i dwie $(n \times n)$ wymiarowe macierze $A=[a_{i,k}]$, $B=[b_{j,l}]$. Należy znaleźć permutację $\Pi=(\Pi(1),...,\Pi(n))$ elementów zbioru N, która minimalizuje funkcje celu $f(\Pi)$ o postaci:

$$f(\Pi) = \sum_{i=1}^{n} \sum_{k=1}^{n} a_{i,k} b_{\Pi(i),\Pi(k)}$$

W terminologii alokacji obiektów zbiór N jest zbiorem numerów obiektów, a $\Pi(i) \in N$, $i=1,...,n$ określa numer obiektu przydzielonego do pozycji i. Macierz A jest wtedy macierzą odległości pomiędzy pozycjami rozmieszczenia obiektów, podczas gdy macierz B opisuje powiązania występujące pomiędzy obiektami. Natomiast funkcja celu $f(\Pi)$, $\Pi \in \Psi$, gdzie Ψ jest zbiorem n-elementowych permutacji, określa koszt globalny eksploatacji systemu.

2. Ewolucyjny proces przeszukiwania przestrzeni rozwiązań

Algorytmy ewolucyjne *(EA)* są niezwykle obiecującą metodą poszukiwania rozwiązań, mającą korzenie w genetyce oraz naturalnej selekcji [4]. Dzięki swoim cechom dają one możliwość przeszukiwania wielkich przestrzeni rozwiązań, wobec których klasyczne metody optymalizacji okazują się bezradne. Algorytmy ewolucyjne powstały z relaksacji założeń i wprowadzania „innowacji" w klasycznych algorytmach genetycznych *(AG)*, które zostały zaproponowane w 1975 roku przez Hollanda jako ogólnego przeznaczenia metaheurystyka, odwołująca się do praw ewolucji [6].

Zastosowanie technik EA w stosunku do konkretnego zadania wymaga spełnienia niewielu prostych warunków, tj.:

* określenia reprezentacji rozwiązania,

* określenia zbioru pseudogenetycznych operatorów (zwanych też krótko operatorami genetycznym),
* określenia funkcji oceny (przystosowania) rozwiązań (ang. *fitness function*).

Dziedziczona wiedza zgromadzona w procesie poszukiwania rozwiązania jest kodowana w postaci listy rozwiązań zapisanej w zbiorze *P*, zwanym dalej populacją, gdzie $M = |P|$ jest rozmiarem populacji. Sam proces ewolucji populacji P, modelowany przez algorytmy ewolucyjne, realizowany jest przez użycie operatorów genetycznych. Każdy operator genetyczny generuje nowe rozwiązania (zwane potomkami), bazując na poprzednich rozwiązaniach (zwanych rodzicami).

W zadaniach permutacyjnych najczęściej stosowana w *GA* i *EA* binarna reprezentacja rozwiązania nie wnosi żadnych ułatwień od strony algorytmu i wymaga zastosowania specjalnego algorytmu poprawy rozwiązań. Wynika to z faktu, że zmiana jedynie jednego bitu w rozwiązaniu może prowadzić do niedopuszczalności rozwiązania, tzn. rozwiązanie przestaje być permutacją. Dlatego też zastosowaliśmy w badanym przez nas algorytmie naturalną reprezentację rozwiązania QAP, permutację, operatory genetyczne zapewniające dopuszczalność otrzymanych rozwiązań-potomków oraz funkcję oceny rozwiązań $f(\Pi)$.

Michalewicz w swojej pracy (monografii [7]) przedstawił zmodyfikowaną wersję algorytmu genetycznego, zwaną *modGA*. Modyfikacja algorytmu *modGA*, w odniesieniu do klasycznego *AG*, polega na tym, że nowy zbiór *P* jest formowany w następujący sposób: tylko część $r < M$ rozwiązań 'starego' zbioru *P* jest wybierana jako rodzice do przekształcenia oraz *r* rozwiązań do odrzucenia. Selekcja jest realizowana zgodnie z wartością funkcji przystosowania. Rozwiązania z wartością większą niż średnia wartość funkcji przystosowania rozwiązań zbioru *P* mają większą szansę zostania rodzicami, natomiast rozwiązania z wartością funkcji przystosowania mniejszą niż średnia mają większą szansę być wybrane do odrzucenia. Nowy zbiór *P* składa się z *(M-r)* rozwiązań starego zbioru oraz *r* nowych rozwiązań-potomków otrzymanych w wyniku przekształcenia przez operatory genetyczne r rodziców wybranych ze starego zbioru *P*. W ten sposób w jednej iteracji tylko *r* $(r \leq M)$ rozwiązań podlega procesowi selekcji i przekształcania, dzięki czemu algorytm *modGA* należy do klasy *Steady State GA*.

W artykule przedstawiamy algorytm *EVOL-2*, gdzie podczas jednej iteracji jest wybierany jeden operator w sposób przypadkowy (rozkład normalny), a następnie *r*, $r \in \{1,2\}$ rozwiązań jest selekcjonowanych z populacji i poddawane przekształceniu. Jeśli został

wybrany operator binarny, wtedy selekcjonujemy dwa rozwiązana, jedno - jeśli unarny. Powyższy schemat zachowania się algorytmu pozwala zakwalifikować go do klasy algorytmów *Steady State GA*.

Założono także, że zbiór P jest liniowo uporządkowany za pomocą funkcji przystosowania: pierwsze rozwiązanie zbioru jest rozwiązaniem najlepszym $\Pi_{best,...}$, ostatnie rozwiązanie (rozwiązanie nr M) jest rozwiązaniem najgorszym Π_{worst} w zbiorze.

Przedstawiona metoda poszukiwania rozwiązania posiada pewną wadę - najlepsze rozwiązania w populacji mają wielokrotnie większą szansę selekcji na rodziców niż pozostałe, co powoduje powstanie tzw. „superosobników". Lepsze rozwiązania mają większą liczbę potomków, a ponieważ populacja ma stały rozmiar, powoduje to eliminację innego niż reprezentowany przez „superosobniki" (być może bardziej obiecującego) „materiału chromosomowego". Efektem tego jest eliminacja w kilku pokoleniach (przebiegach algorytmu) z populacji innych rozwiązań - populacja staje się bardzo jednorodna, czego wynikiem jest utknięcie procesu optymalizacji w minimum lokalnym.

Aby zabezpieczyć algorytm przed wpadaniem w minima lokalne, zaproponowano wzbogacenie mechanizmu selekcji naturalnej o tzw. „tabu operatorowe". Mechanizm ten działa na zasadzie zakazu stosowania operatora genetycznego w stosunku do potomków, którzy powstali pod jego wpływem. W praktycznym zastosowaniu wymaga to wprowadzenia dla każdego rozwiązania dodatkowego parametru, określającego jego „pochodzenie". Jest to swego rodzaju dodatkowa forma pamięci, charakteryzująca całą populację. W praktycznych aplikacjach można natknąć się na stosowanie podobnych mechanizmów, lecz najczęściej odnoszą się one np. do zakazu krzyżowania rodziców z potomkami itp., a nie do stosowanych operatorów.

Informacja zawarta w populacji jest przetwarzana w specyficzny sposób przez każdy operator. W przedstawionym algorytmie zmodyfikowano metodę przetwarzania informacji podczas procesu poszukiwania najlepszego rozwiązania. Zastosowanie „tabu operatorowego" umożliwia poprawę „wymiany informacji" pomiędzy operatorami, zapobiegając jej „jednotorowemu" przetwarzaniu. W efekcie prowadzi to do największej dywersyfikacji przestrzeni i kierunków poszukiwań algorytmu.

Algorytm po wyborze $r \in \{1,2\}$ rodziców i odpowiedniego operatora tworzy potomków. Tak powstali potomkowie zapamiętują swoje „pochodzenie" i otrzymują zakaz wchodzenia w „związki" z innymi rozwiązaniami poprzez operator, który je stworzył. Zakaz

ten obowiązuje pewną określoną liczbę prób użycia tego rozwiązania przez ten operator, dopuszczając jednocześnie użycie go przez inne operatory. Długotrwałość tego zakazu jest parametrem algorytmu.

Poszukując efektywnych algorytmów aproksymacyjnych dla *NP-trudnych* problemów optymalizacji kombinatorycznej oraz nowych metod, które można stosować w tych algorytmach, przebadano metodę umożliwiającą zwiększenie ich efektywności. Metoda ta nie wymaga zwiększenia liczby parametrów algorytmu i może zostać zastosowana w prosty sposób jako jeden ze składników algorytmów ewolucyjnych, wspomagając proces przeszukiwania przestrzeni rozwiązań.

Właściwości tej metody zostały przebadane na przykładzie algorytmu ewolucyjnego, charakteryzującego się większą liczbą zastosowanych operatorów genetycznych, co wydaje się być jedynym warunkiem jej efektywnego zastosowania.

3. Algorytm

EVOL-2 wykorzystuje dwa operatory unarne i trzy operatory krzyżowania. Zauważmy, że w przypadku formalizacji rozwiązania zagadnienia w postaci permutacji Π informacją zawartą w rozwiązaniu jest tylko *kolejność* elementów. W takim przypadku w algorytmie genetycznym można zastosować tylko operatory zmieniające kolejność elementów permutacji. W przeprowadzonych badaniach komputerowych zastosowano dwa unarne operatory genetyczne. Do określenia jednego potomka Π^1 na podstawie rodzica $\Pi = (\Pi(1), \Pi(2),..., \Pi(i),..., \Pi(j),..., \Pi(n))$ należy wykonać następujące operacje:

1. Operator mutacji RM

Wybierz w sposób losowy (rozkład równomierny) dwa elementy permutacji Π, powiedzmy $\Pi(i)$ i $\Pi(j)$, i dokonaj zamiany ich położenia wyznaczając w ten sposób potomka $\Pi^1 = (\Pi(1),..., \Pi(i-1), \Pi(j),..., \Pi(j-1), \Pi(i),..., \Pi(n))$.

2. Operator optymalizacji lokalnej LO

(a). Określ otoczenie $S(\Pi)=\{\Pi': \Pi'=(\Pi(1),..., \Pi(i-1), \Pi(j),..., \Pi(j-1), \Pi(i),..., \Pi(n)), j\neq i\}$ permutacji $\Pi=(\Pi(1),..., \Pi(i-1), \Pi(i),..., \Pi(j-1), \Pi(j),..., \Pi(n))$;

(b). Znajdź permutację $\Pi' = arg\ min\{f(\hat{\Pi}): \hat{\Pi} \in S(\Pi), \hat{\Pi} \neq \Pi\}$, gdzie $f(\Pi)$ oznacza funkcję oceny rozwiązania Π;

(c). Jeżeli $f(\Pi') < f(\Pi)$ to podstaw $\Pi := \Pi'$ i przejdź do instrukcji **(a)**, w przeciwnym przypadku podstaw $\Pi^1 := \Pi$ i powróć do programu głównego.

Zauważmy, że operator mutacji nie zmienia wartości elementów rozwiązania a tylko kolejność elementów w rozwiązaniu. Podobnie otoczenie *S(Π)* jest definiowane przez zamianę kolejności elementów rozwiązania Π. Przypomnijmy, że siła ewolucyjnego poszukiwania rozwiązania ma swe źródło w przypadkowej rekombinacji materiału genetycznego rodziców. Proces rekombinacji jest realizowany przez operatory (binarne) krzyżowania.

Dla rozwiązań, w których ważna jest kolejność elementów, operatory krzyżowania są bardziej skomplikowane, ponieważ klasyczne (ślepe) krzyżowanie może prowadzić do wyznaczenia rozwiązań niedopuszczalnych. Na przykład stosując jednopunktowy klasyczny operator do rodziców-permutacji:

Π_1 = (1,2,3,4,5,6,7,8,9,10), Π_2= (7,3,1,10,9,2,5,4,8,6) otrzymujemy (jeżeli punkt krzyżowania jest po 5 elementach) potomków Π^1 = (1,2,3,4,5,2,5,4,8,6),

Π^2 = (7,3,1,10,9,6,7,8,9,10), z których żaden nie jest rozwiązaniem dopuszczalnym, tj. permutacją. Tak więc należy zastosować ściśle określone heurystyczne operatory krzyżowania. W naszych eksperymentach komputerowych stosowaliśmy trzy operatory krzyżowania:

3. Operator PMX (partially matched crossover- zob., [4] str.172-174)

4. Operator OX (order crossover- zob., [4] str.172-174)

5. Operator COMPX (composition crossover - zob., [10] str. 451-458) .

ALGORYTM EVOL2:

Aby wyznaczyć rozwiązanie przybliżone Π_{approx} wykonaj następującą procedurę:

Krok 1. Wyznaczenie populacji początkowej

Wygeneruj w sposób losowy *M* permutacji Π oraz określ funkcje przystosowania *f(Π)* dla każdej z nich. Z wygenerowanych rozwiązań utwórz populację początkową P o rozmiarze *M*, uporządkowaną wg wartości funkcji przystosowania *f(Π)*, tzn. permutacja nr 1 jest najlepsza ($\Pi_{best} = \arg\min\{f(\Pi):\Pi \in P\}$) i ostatnia permutacja nr *M* jest najgorsza ($\Pi_{worst} = \arg\max\{f(\Pi):\Pi \in P\}$). Dla każdego z rozwiązań ustaw wartość parametru T=0.

Krok 2. Wybór operatora genetycznego

Wylosuj typ operatora genetycznego ze zbioru {*RM, LO, OX, PMX, COMX*}, gdzie każdy z operatorów jest losowany z prawdopodobieństwem $p_i \geq 0$, $i \in \{RM, LO, OX, PMX, COMX\}$, przy czym $p_{RM} + p_{LO} + p_{OX} + p_{PMX} + p_{COMX} = 1$.

Krok 3. Wybór rodziców rodzica

Dla wybranego operatora wylosuj, zgodnie z rozkładem równomiernym, ze zbioru *P*, zależnie od typu operatora, jedno (w przypadku operatora unarnego) lub dwa (w przypadku operatora krzyżowania) rozwiązania, które nazywamy rodzicami.

Krok 4. Sprawdzenie warunku dopuszczalności zastosowania operatora

Jeśli jedno z wybranych rodziców powstało w wyniku zastosowania aktualnie wylosowanego operatora genetycznego i wartość parametru T>0, to dekrementuj wartość T oraz idź do kroku 3. W pp. idź do kroku 5.

Krok 5. Generowanie potomków

Za pomocą wylosowanego operatora genetycznego dokonaj modyfikacji rozwiązania-rodzica/rozwiązań-rodziców i wyznacz w ten sposób rozwiazanie-potomka/rozwiązania-potomków. Zapisz dla każdego z potomków, w wyniku działania jakiego operatora powstał oraz ustaw $T = T_{ALG}$.

Krok 6. Poprawa rozwiązań w zbiorze P

Dla każdego potomka oblicz wartość funkcji przystosowania. Jeżeli ta wartość jest lepsza od wartości najgorszego rozwiązania zbioru P, to umieść takiego potomka w zbiorze P, usuwając zarazem z tego zbioru rozwiązanie najgorsze.

Krok 7. Warunek STOP'u

Jeżeli wygenerowano zadaną liczbę L potomków, to STOP; zwróć najlepsze rozwiązanie zbioru P-$\Pi_{approx} = arg\ min\ \{f(\Pi): \Pi \in P(O)\}$. W przeciwnym wypadku idź do kroku 2.

Wielkości takie, jak: rozmiar populacji M, liczba iteracji algorytmu L, długotrwałość zakazu zastosowania „rodzicielskiego" operatora do rozwiązania T_{ALG} i prawdopodobieństwa selekcji operatorów RM - p_{RM}, LO - p_{LO}, OX -p_{OX}, PMX - p_{PMX}, $COMX$ - p_{COMX} są parametrami algorytmu *EVOL-2*. Parametr T_{ALG} można określić także jako „długotrwałość tabu operatorowego" regulującego współdziałanie operatorów.

4. Wyniki badań komputerowych

Na podstawie algorytmu *EVOL-2* wykonano eksperymentalny program komputerowy, zakodowany w języku C++, dla komputerów kompatybilnych z IBM PC, pracujących pod systemem MS DOS. W tej części przedstawiamy wyniki badań eksperymentalnych uzyskanych za pomocą stworzonego systemu komputerowego dla zbioru 34 testów o rozmiarze n=26-60, zaczerpniętych z biblioteki QAPLIB-A [2]. Wszystkie eksperymenty komputerowe wykonane na postawie algorytmu EVOL-2 korzystają z następującego zbioru parametrów: L=7000 iteracji, M - 100 (rozmiar populacji), $p_{RM} = 0.2$, $p_{LO} = 0.02$, $p_{OX} = 0.2$, $p_{PMX} = 0.2$, $p_{COMX} = 0.38$, $T_{ALG} \in \{0, 1, 2, 3\}$.

W tabeli 1 używamy następującego nazewnictwa:

T_{ALG} - długotrwałość zakazu zastosowania „rodzicielskiego" operatora do rozwiązania;

$f_{QAPLIB\text{-}A}$ - najlepsza znana wartość funkcji przystosowania zaczerpnięta z biblioteki QAPLIB-A,

f_{start} - wartość funkcji przystosowania najlepszego rozwiązania w populacji startowej

$\Pi_{start}=arg\ min\ \{f(\Pi):\Pi \in P(O)\}$,

f_{ap} - wartość funkcji przystosowania najlepszego znalezionego rozwiązania dla Π_{approx},

$E = 100\%(f_{ap} - f_{QAPLIB\text{-}A})/f_{QAPLIB\text{-}A}$,

L_{ap} - liczba iteracji po ilu zostało znalezione najlepsze rozwiązanie Π_{approx},

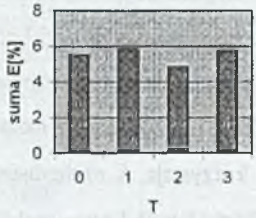
t_{ap} - czas wykonania w [ms], L_{ap} iteracji algorytmu, po których zostało znalezione najlepsze rozwiązanie,

t_{ex} - czas wykonania w [ms], L = 7000 iteracji algorytmu.

Na podstawie eksperymentów komputerowych (zob. tabela 1) możemy stwierdzić, że najwięcej najlepszych rozwiązań uzyskano dla wartości parametru T_{ALG} =3. Z drugiej jednak strony średnia jakość rozwiązań (zob. rys 1) jest najlepsza dla wartości T_{ALG} = 2.

5 .Podsumowanie

Należy stwierdzić, że algorytm *EVOL-2* jest algorytmem przeznaczonym do rozwiązywania szerokiej gamy zadań permutacyjnych, a nie specjalizowaną procedurą ukierunkowaną na rozwiązywanie jedynie problemu *QAP*. Zaproponowana metoda „tabu operatorowego" może być doskonałym uzupełnieniem innych procedur z dziedziny algorytmów ewolucyjnych.



Rys. 1
Fig.1

LITERATURA

1. Brown D.E., Huntley Ch.L., Spollane A.R.: A parallel genetic heuristic for the quadratic assigment problem. Proc. of the Third Int. Conference on Genetic Algorithms, Georg Manson Univ., 1989, pp. 406-415.

2. Burkard R.E., Karisch S.E., Rendl F., QAPLIB-A Quadratic Assignment Problem Library, European Journal of Operational Resarch, 55, 1991, pp.115-119.

3. Brukard R.E., Stratmann K.H.: Numerical investigation on quadratic assigment problems. Naval Research Logistics Quaterly, vol.25, 1978, pp.129-147.

4. Filipowicz B., Wala K.: Algorytmy optymalizacji kwadratowego zagadnienia przydziału. Kwartalnik Elektrotechnki, z.1, 1992, Wydawnictwo AGH w Krakowie.

5. Finke G., Burkard R.E, Rendl F.: Quadratic assigments problems. Annals of Discrete Mathematics, vol.31, North-Holland, 1987, pp. 61 -82.

6. Holland J., Adaptation in natural and artificial systems. Univ. of Michigan Press, Ann Arbor, MI, 1975.

7. Michalewicz Z., Genetic algorithms + data structures = evolution programs, Springer-Verlag, Berlin, 1992.

8. Michalewicz Z., Heuristic Methods for Evolutionary Computation Techiques, Journal of Heuristics, 1, Kluwer Academic Publishers, 1995, pp. 177-206.

9. Wala K., Chmiel W., An improved genetic algorithm for *NP-hard* permutation problems, Proc. of Third Int. Symposium on Methods and Models in Automation and Robotics, 10-13 September 1996, Międzyzdroje, Poland, Vol.3, pp. 1163-1166.

10 Wala K., Chmiel W., Investigations of crossover genetic operators for permutational optimization problem, University of Mining and Metallurgy Press, Elektrotechnika 14,2, Kraków, 1993, pp. 451-458.

11 Wala K., Chmiel W., Evolution Algorithm for Quadratic Assignment Problem, University of Mining and Metallurgy Press, Elektrotechnika 1,1, Kraków 1997, pp. 409-414.

Recenzent: Dr hab.inż. Adam Janiak, prof.Pol.Wroc.

Abstract

Permutation problems are an important class of decision problems in the combinatorial optimization domain. Classical instances of permutation problems include quadratic assignment problem *(QAP)*, graph coloring, production scheduling problems, as well as variety of design problems. The paper presents the results of computer investigation of approximate algorithm, realized evolution artificial search process, for *QAP* as a hard instance of permutation problems and, on the other hand, there is a rich Quadratic Assignment Problem Library, called QAPLIB-A, with test task of this problem for approximate algorithm examinations. *QAP* generalizes many NP-hard combinatorial optimization problems, including the travelling salesman problem, and until now even quite small instances for exact algorithm are computationally intractable. Evolutionary algorithms (*EAs*) are powerful search techniques taking inspiration from genetics and natural selection. They can effectively explore very large solutions spaces without being trapped by local optima. The one requirements for applying EAs to the problem are: solution

representation of the problem to be solved, a set of pseudo-genetic operators called briefly genetic operators and a evaluation function of the solution called fitness function.

We examine evolution search process called *EVOL-2*, where in course of one iteration initially one genetic operator is randomly chosen and then r, $r \in \{1,2\}$, solutions are selected from the population and processed: one solution if unary operator is chosen and two in case of crossover operator. The additional restriction for the choice of parent basing on the origin, were introduce in the algorithm. Thus the algorithm also belongs to the class of *Steady State GAs* and in this way it has all features of the *modGA*.