ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1997
Seria: INFORMATYKA z. 33 Nr kol.1377

Anna TOMASZEWSKA
Politechnika Śląska, Instytut Informatyki

ROZWIĄZYWANIE PROBLEMU ROZBICIA ZBIORU ZA POMOCĄ ALGORYTMÓW GENETYCZNYCH

Streszczenie. Zagadnienie rozbicia zbioru jest przykładem złożonej optymalizacji kombinatorycznej i należy do grupy problemów NP-zupełnych. Ponieważ do rozwiązywania innych problemów tej klasy znalazły zastosowanie algorytmy genetyczne, w pracy, której dotyczy to opracowanie, podjęto próbę adaptacji mechanizmów genetycznych również dla zagadnienia rozbicia zbioru. Prezentowane są wyniki badań wpływu wartości wybranych parametrów algorytmu na wyniki jego działania.

RESOLVING THE SET PARTITIONING PROBLEM WITH USE OF GENETIC ALGORITHMS

Summary. The Set Partitioning Problem (SPP) is a difficult combinatorial optimization problem that belongs to NP-complete problems group. Genetic Algorithms (GA) proved their efficiency in searching for approximate solutions for other problems of this class. In this paper a trial of adaptation of GA for SPP is presented. The results of research concerning the influence of some GA parameters' values on process of searching the solution and its effects are described as well.

1. Wprowadzenie

Problem rozbicia zbioru, który wraz z pokrewnymi problemami pokrycia i upakowania tworzy grupę zagadnień pokrycia, należy do klasy tzw. problemów NP-zupełnych, tj. takich, dla których nie jest znany deterministyczny algorytm o złożoności wielomianowej znajdujący rozwiązanie optymalne. Istnieje kilka sposobów poszukiwania najlepszego rozwiązania dla tego problemu, charakteryzują się one jednak wykładniczą średnią czasową złożonością obliczenio-

wą i wykładniczą zajętością pamięci. W związku z tym tracą zastosowanie dla zbiorów bazowych o dużej mocy.

W pracy, której dotyczy to opracowanie, zdecydowano się na zgoła inne podejście do rozważanego problemu. Ponieważ w praktycznych zastosowaniach zagadnienia rozbicia zbioru najczęściej nie jest konieczne znalezienie rozwiązania najlepszego (optymalnego), a jedynie wystarczająco dobrego, podjęto poszukiwania rozwiązania problemu wśród algorytmów dających rozwiązania prawie optymalne. Z opublikowanych wyników prowadzonych na świecie badań wynika, że algorytmy genetyczne znalazły zastosowanie do rozwiązywania wielu problemów optymalizacji dyskretnej [9,10]. Ponieważ podobnie jak problemy upakowania i pokrycia, również rozbicie zbioru należy do problemów optymalizacji dyskretnej, postanowiono podjąć próbę zaadaptowania mechanizmów dziedziczenia i doboru naturalnego dla tego zagadnienia licząc na to, że algorytm zbudowany w ten sposób umożliwi znajdowanie dobrych rozwiązań.

2. Sformułowanie problemu rozbicia zbioru

Niech dany będzie zbiór bazowy: $N=\{1,2,\ldots,n\}$ oraz rodzina podzbiorów tego zbioru $P=\{P_j\}_{j\in M}$ gdzie $M=\{1,2,\ldots,m\}$.

Zdefiniujmy tzw. *funkcję celu* przyjmująca wartości nieujemne dla każdego podzbioru zbioru bazowego N:

$$\bigwedge_{P_j \in P} F_c(P_j) = c_j \geq 0, \tag{1}$$

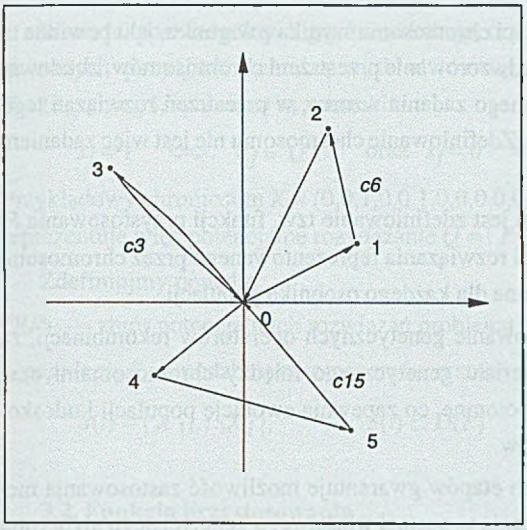
Rozbicie zbioru N definiujemy jako podrodzinę Q rodziny P, której wszystkie elementy są rozłączne i pokrywają wszystkie elementy zbioru N [8].

Funkcja celu dla rozbicia zbioru jest zdefiniowana jako suma wartości funkcji kosztów dla podzbiorów wchodzących w skład rozbicia:

$$F_c(Q) = \sum_{i=1}^{k} F_c(Q_j) \tag{2}$$

gdzie $Q_j \in P$. Problem rozbicia zbioru polega, najogólniej rzecz ujmując, na znalezieniu rozbicia Q o minimalnej wartości funkcji celu.

Szczególnym przypadkiem problemu rozbicia zbioru jest zagadnienie dostawy, gdzie zbiór bazowy N reprezentuje zbiór klientów (miast, węzłów), którzy mają być obsłużeni przez bazę-magazyn (rys. 1). Punkty 1, 2, 3, 4 i 5 reprezentują klientów, zaś punkt 0 - magazyn. Zakładamy, że magazyn dysponuje nieograniczoną liczbą ciężarówek, służących do rozwożenia towarów do klientów. Ograniczona jest jedynie liczba klientów, którzy mogą być obsłużeni



Rys.1. Rozbicie zbioru na przykładzie problemu dostawy
Fig.1. Delivery problem – an example of SPP

w ramach jednej trasy; liczbę tę oznaczamy k. Każdej możliwej trasie obejmującej od 1 do k klientów przyporządkowany jest koszt wynikający z odległości i oznaczony c_j, gdzie j jest numerem trasy w macierzy incydencji przedstawionej w tabeli 1 poniżej.

Element macierzy incydencji przedstawionej w tabeli 1 $a_{ij} = 1$ wtedy i tylko wtedy, gdy trasa P_j (określona przez kolumnę) pokrywa element i (określony przez wiersz) zbioru bazowego, w przeciwnym przypadku $a_{ij} = 0$.

Zadanie polega na znalezieniu takiej kombinacji tras, której koszt będzie minimalny, a każdy element zbioru bazowego (węzeł) pokryty będzie jednokrotnie.

Tabela 1

Macierz incydencji $A = [a_{ij}]$ dla n=5 i k=3

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}	P_{17}	P_{18}	P_{19}	P_{20}	P_{21}	P_{22}	P_{23}	P_{24}	P_{25}
1	1					1	1	1	1							1	1	1	1	1	1				
2		1				1				1	1	1				1	1	1				1	1	1	
3			1				1			1			1	1		1			1	1		1	1		1
4				1				1			1		1		1		1		1		1	1		1	1
5					1				1			1		1	1			1		1	1		1	1	1
Fc	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}	c_{21}	c_{22}	c_{23}	c_{24}	c_{25}

3. Definicja podstawowych elementów algorytmu genetycznego dla problemu rozbicia zbioru

Zastosowanie algorytmu genetycznego do konkretnego problemu uwarunkowane jest możliwością znalezienia odpowiedniej reprezentacji tego problemu, a właściwie jego rozwiązania, nadającej się do implementacji w języku programowania. Reprezentację tę nazywa się chromosomem i to ona podlega przetwarzaniu w procesie poszukiwania rozwiązania.

Problem znalezienia odpowiedniej postaci chromosomu wynika z warunku, jaki powinna ta postać spełniać [3]. Warunek ten mówi, iż odwzorowanie przestrzeni chromosomów, zbudowanych według pewnego, właściwego dla danego zadania wzorca, w przestrzeń rozwiązań tego zadania musi być wzajemnie jednoznaczne. Zdefiniowanie chromosomu nie jest więc zadaniem trywialnym.

Kolejnym ważnym elementem projektu jest zdefiniowanie tzw. funkcji przystosowania F_f (*ang. fitness function*) będącej miarą jakości rozwiązania reprezentowanego przez chromosom. Jej wartości muszą być nieujemne i określone dla każdego osobnika populacji.

Trzeci, równie ważny etap, to zdefiniowanie genetycznych operatorów rekombinacji, za pomocą których następuje wymiana materiału genetycznego między chromosomami oraz przekształcanie pokoleń rodzicielskich w potomne, co zapewnia ewolucję populacji i udoskonalanie wchodzących w ich skład osobników.

Dopiero pomyślne przejście tych trzech etapów gwarantuje możliwość zastosowania mechanizmów doboru naturalnego i dziedziczenia w postaci właściwego genetycznego algorytmu poszukiwania rozwiązania.

3.1. Postać chromosomu

Inspiracją do zaprojektowania postaci chromosomu była macierz incydencji przedstawiona w tabeli 1, opisująca pokrycia zbioru bazowego przez poszczególne trasy.

Zdecydowano się określić *chromosom* jako ciąg bitowy o długości równej liczbie możliwych tras (m) dla danych n (moc zbioru bazowego) i k (maksymalna liczba klientów, którzy mogą być obsłużeni w ramach jednej trasy). Wielkość ta określona jest wzorem:

$$m = \sum_{i=1}^{k} \binom{n}{i} \tag{3}$$

i wynosi np. dla $n = 5$ i $k = 3$ $m = 25$

$n = 6$ i $k = 3$ $m = 41$

$n = 7$ i $k = 3$ $m = 63$

$n = 8$ i $k = 3$ $m = 92$ itd.

Alfabet genetyczny to zbiór dwuelementowy o wartościach 0 i 1 ($B = \{0, 1\}$). *Genami* są poszczególne bity ciągu będącego chromosomem, przyjmują one wartości z alfabetu genetycznego. Formalnie więc chromosom zapisać można jako wektor:

$$X = (x_1, \ldots, x_m), \text{ taki że: } \bigwedge_{j \in \langle 1, m \rangle} x_j \in B,$$

Wartości poszczególnych genów wskazują, czy trasy znajdujące się w macierzy incydencji na danych pozycjach wchodzą w skład rozwiązania, czy nie. Tak więc dla chromosomu $X = (x_1, \ldots, x_m)$, spełnione są warunki:

$$x_j = 1 \quad <=> \quad P_j \in Q \qquad \text{oraz} \quad x_j = 0 \quad <=> \quad P_j \notin Q$$

Przykładowy chromosom $X = (0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0)$ reprezentuje więc potencjalne rozwiązanie $Q = \{P_3, P_6, P_{15}\}$

Zdefiniujmy ponadto:

$D(P)$ — zbiór potencjalnych rozwiązań problemu (każdy $X = (x_1, \ldots, x_m) \in D(P)$)

i *populację* S — podzbiór chromosomów o stałej dla algorytmu liczności (l).

$$S(t) = \{X_1^t, \ldots, X_l^t\}, \qquad S(t) \subset D(P) \qquad \text{– populacja w pokoleniu t.}$$

3.2. Funkcja przystosowania

Aby określić funkcję przystosowania, należy rozpocząć od zdefiniowania pewnych wielkości. Celem poszukiwań jest znalezienie rozbicia zbioru bazowego o minimalnym koszcie sumarycznym, określonym wzorem:

$$Z = \sum_{j=1}^{m} c_j x_j \tag{4}$$

gdzie c_j jest kosztem trasy j-ej, a x_j wartością j-tego bitu chromosomu. Rozwiązanie jest rozbiciem (rozwiązaniem akceptowalnym) wtedy i tylko wtedy, gdy

$$\sum_{j=1}^{m} a_{ij} x_j = 1 \qquad \text{dla każdego} \qquad i = 1..n \tag{5}$$

gdzie x_j określone jest jak we wzorze (4), a a_{ij} jest elementem macierzy incydencji A.

Funkcja celu, określająca koszt sumaryczny rozwiązania reprezentowanego przez chromosom, zdefiniowana została jako

$$F_c(X) = \sum_{j=1}^{m} c_j x_j + \sum_{i=1}^{n} H_i \left(\sum_{j=1}^{m} a_{ij} x_j - 1 \right)^2, \tag{6}$$

gdzie pierwszy składnik sumy jest rzeczywistym kosztem potencjalnego rozwiązania, składnik drugi natomiast stanowi tzw. "koszt karny" dla rozwiązań nieakceptowalnych, czyli takich, które nie są rozbiciem, a jedynie pokryciem, lub wogóle nie pokrywających zbioru bazowego).

H_i jest wielkością różną dla każdego z węzłów i zależną od wartości λ_i zdefiniowanej jako średnia wartość kosztów tras pokrywających węzeł i-ty. Zależność H_i od λ_i może być stała

lub zmienna. Uwarunkowane to może być na przykład brakiem pokrycia węzła lub pokryciem wielokrotnym. Na podstawie przeprowadzonych niezależnie badań dla eksperymentów prowadzonych w ramach prezentowanej pracy przyjęto $H_i = \lambda_i$ dla braku pokrycia węzła i $H_i = \lambda_i^2$ dla pokrycia wielokrotnego.

Funkcja przystosowania F_f powinna być tak zdefiniowana, aby spełniony był warunek:

$$\bigwedge_{X_i^t \in S(t)} F_f(X_i^t) \geq 0 \tag{7}$$

przy czym zgodnie z definicją funkcji przystosowania dla klasycznego algorytmu genetycznego wartości tej funkcji powinny być wprost proporcjonalne do jakości rozwiązania reprezentowanego przez chromosom, dla którego jest określona [3].

Funkcja celu, która została wcześniej zdefiniowana, nie spełnia ostatniego założenia, ponieważ jej wartości są odwrotnie proporcjonalne do jakości rozwiązania. Im niższa jej wartość, tym lepsze rozwiązanie. W celu usunięcia tej sprzeczności wprowadzono przekształcenie odwrotne funkcji celu i określono funkcję przystosowania jako:

$$F_f(Q) = \frac{1}{F_c(Q)} \tag{8}$$

3.3. Genetyczne operatory rekombinacji

Genetyczne operatory rekombinacji, zwane również operatorami genetycznymi, umożliwiają wymianę materiału genetycznego w ramach populacji i między nimi. Dla projektowanego algorytmu genetycznego operatory zdefiniowano w sposób przedstawiony poniżej.

REPRODUKCJA - operacja powielania osobników, polega na skopiowaniu chromosomu w nie zmienionej postaci do kolejnej populacji. Formalnie można zapisać reprodukcję jako przekształcenie $R: D(P) \rightarrow D(P)$ takie, że:

$$R(X_i^t) = X_i^{t+1}, \text{ gdzie } X_i^t = X_i^{t+1} = (x_1, \ldots, x_m).$$

Selekcja chromosomów do reprodukcji odbywa się metodą ruletki, w której prawdopodobieństwo wylosowania osobnika jest wprost proporcjonalne do wyznaczonej dla niego wartości funkcji przystosowania. Dzięki tej metodzie selekcji, liczność osobników lepiej przystosowanych rośnie w kolejnych populacjach, gorzej przystosowane natomiast giną.

KRZYŻOWANIE - operacja polegająca na wymianie fragmentów chromosomów między sobą. Biorą w niej udział zawsze dwa osobniki losowo wybrane z populacji, niezależnie od wartości funkcji przystosowania. Na potrzeby algorytmu zdefiniowano tę operację jako krzyżowanie jednopunktowe, polegające na wymianie między chromosomami ich fragmentów końcowych, począwszy od genu o numerze $s+1$, gdzie s losowane jest przy każdej operacji.

Formalnie krzyżowanie zapisać można jako

przekształcenie $K_s: D(P) \times D(P) \rightarrow D(P) \times D(P)$

takie, że: $K_s(X_i^t, X_j^t) = (X_i^{t+1}, X_j^{t+1})$,

gdzie $X_i^t = (x_{i1}, \ldots, x_{is}, x_{is+1}, \ldots, x_{im})$, $X_j^t = (x_{j1}, \ldots, x_{js}, x_{js+1}, \ldots, x_{jm})$,

$X_i^{t+1} = (x_{i1}, \ldots, x_{is}, x_{js+1} \ldots, x_{jm})$ $X_j^{t+1} = (x_{j1}, \ldots, x_{js}, x_{is+1} \ldots, x_{im})$

np.: K_7: (0,0,1,0,1,1,0,$\underline{1,0,1}$) → (0,0,1,0,1,1,0,$\underline{1,1,0}$)

(1,1,0,1,0,1,0,$\underline{1,1,0}$) → (1,1,0,1,0,1,0,$\underline{1,0,1}$)

MUTACJA - operacja polegająca na wymianie s-tego, losowo wybranego, genu chromosomu. Ponieważ geny przybierają wartości z alfabetu dwuelementowego, to mutację zdefiniowano jako inwersję prostą - tzw. flipping - polegającą na zanegowaniu wartości wylosowanego do zmutowania genu. Formalnie zdefiniowano mutację jako:

przekształcenie $M_s: D(P) \rightarrow D(P)$ takie, że:

$M_s(X_i^t) = X_i^{t+1}$, gdzie $X_i^t = (x_1, \ldots, x_s, \ldots, x_m)$, $X_i^{t+1} = (x_1, \ldots, \overline{x_s}, \ldots, x_m)$

s - numer zmutowanego genu

np.: M_7: (0,1,0,0,1,1,$\underline{1}$,0,1,0) → (0,1,0,0,1,1,$\underline{0}$,0,1,0)

W systemie poszukiwania rozwiązań zagadnienia optymalizacyjnego bazującym na algorytmie genetycznym zawsze określone są prawdopodobieństwa: p_M - wykonania na chromosomie mutacji oraz p_K - wykonania krzyżowania pary chromosomów, przy czym podobnie jak w naturze, $p_M << p_K$.

4. Opis algorytmu

Dla problemu rozbicia zbioru zaprojektowano algorytm genetyczny o strukturze odpowiadającej algorytmowi klasycznemu typu pokoleniowego.

Poszukiwanie rozwiązania rozpoczyna się od wygenerowania losowej populacji chromosomów. Kolejnym etapem jest wyznaczenie wartości funkcji przystosowania dla każdego chromosomu. Następnie cyklicznie generowane są pokolenia potomne, aż do momentu uzyskania chromosomu o zadowalającej wartości funkcji przystosowania lub osiągnięcia ustalonej wcześniej liczby pokoleń. Cykl powstawania nowego pokolenia obejmuje następujące kroki:

— wyselekcjonowanie metodą ruletki osobnika z populacji rodziców,

— wylosowanie dla niego operacji genetycznej (reprodukcji lub krzyżówki),

— jeżeli wylosowano krzyżówkę, to z populacji rodzicielskiej selekcjonuje się jeszcze jeden chromosom, który będzie stanowił parę dla pierwszego,
— wylosowany chromosom (lub chromosomy) poddawany jest mutacji z prawdopodobieństwem p_M ,
— chromosom (lub para) poddawany jest wybranej uprzednio operacji genetycznej,
— powstałe potomstwo po wyznaczeniu dla niego wartości funkcji przystosowania umieszczane jest w populacji potomnej,
— jeżeli spełniony jest warunek zakończenia poszukiwań, wówczas jako rozwiązanie zwracany jest chromosom o największej wartości funkcji przystosowania.

Każdorazowo na początku iteracji tworzenia nowego pokolenia usuwana jest poprzednia populacja rodziców, a jego miejsce zajmuje wygenerowana w poprzednim kroku populacja potomna.

5. Reprezentacja danych dla algorytmu

Ze względu na charakterystyczną zależność między operacjami, którym podlegają dane, a tymi danymi, polegającą na tym, iż każdemu rodzajowi danych odpowiada specyficzna dla niego operacja lub ich grupa (na przykład: chromosom - mutacja, reprodukcja, populacja - selekcja,wyznaczanie wskaźników jakości dla jej elementów), zdecydowano się zastosować do reprezentacji "materiału genetycznego" obiektowe typy danych jako te, które najlepiej oddają powiązanie przedmiotu operacji z operacją na nim wykonywaną.

Ponadto, ponieważ chromosomy są w przeważającej większości przypadków wektorami rzadkimi, zrezygnowano ze ścisłego odwzorowania modelu chromosomu w ciąg bitowy na rzecz zmieniającej dynamicznie swe rozmiary kolekcji numerów tras wchodzących w skład potencjalnego rozwiązania. Pozwala to na znaczną oszczędność pamięci operacyjnej, zwłaszcza dla zbiorów bazowych o dużej mocy.

6. Przebieg badań

Podstawowym celem badań było sprawdzenie, czy system poszukiwania rozwiązania problemu rozbicia zbioru korzystający z algorytmu genetycznego i zaprojektowany w sposób opisany powyżej pozwala znajdować dobre rozwiązania problemu. Szereg wstępnych eksperymentów wykazał bardzo szybko skuteczność działania algorytmu.

Jednakże ponieważ system jest w wysokim stopniu sparametryzowany, zgodnie z oczekiwaniami zachowanie się algorytmu było bardzo zróżnicowane w zależności od początkowego

ustawienia zmiennych wielkości. W dalszej części badań skupiono się więc głównie na analizie zależności wyników działania algorytmu od wartości jego zmiennych parametrów.

Grupę parametrów systemu stanowią: liczność zbioru bazowego (n), maksymalna liczba węzłów, które mogą być pokryte przez jedną trasę (k), maksymalny koszt jednej trasy (*maxkoszt*), liczność populacji (l), maksymalna liczba pokoleń (it) lub oczekiwana wartość funkcji dopasowania powodująca przerwanie poszukiwań (*ffmax*), prawdopodobieństwo mutacji (p_M), prawdopodobieństwo krzyżówki (p_K), stopień wypełnienia jedynkami chromosomów początkowej populacji rodzicielskiej (*ifr*) oraz zależność H_i od λ_i.

Spośród nich na początek do badań wybrano prawdopodobieństwo mutacji i liczność populacji chromosomów. W celu umożliwienia analizy rezultatów system został przystosowany do prowadzenia eksperymentów przez dodanie modułów śledzenia, rejestracji i przetwarzania wyników. Dodano również mechanizm zapobiegania zbieżności algorytmu do jednego rozwiązania. Część parametrów przyjęto jako stałe w celu umożliwienia analizy porównawczej. Tak więc ustalone zostały przede wszystkim warunki początkowe: parametry n i k, *maxkoszt* oraz wynikające z ostatniego koszty poszczególnych (wszystkich możliwych) tras. Na tym etapie badań określono p_K=1.0, a więc z pokolenia rodziców o liczności l selekcjonowuje się $l/2$ par podlegających krzyżówkom, żaden chromosom nie jest w prosty sposób reprodukowany. Przyjęto model generowania it pokoleń ze śledzeniem rezultatów.

W celu lepszego określenia przebiegu i wyników działania algorytmu zdefiniowano przedstawione poniżej wielkości (współczynniki dobroci):

PzR – procent pokoleń, w których powstały jakiekolwiek rozwiązania akceptowalne (rozbicia),

RR – ilość różnych rozbić zbioru wśród najlepszych przez wszystkie pokolenia,

V – współczynnik różnorodności wśród najlepszych rozwiązań z kolejnych pokoleń,

Rsr – średnia liczba rozbić zbioru w jednym pokoleniu,

K_min – koszt najlepszego uzyskanego chromosomu,

K_sr – średni koszt najlepszych chromosomów z kolejnych pokoleń,

FF_max – wartość funkcji przystosowania najlepszego uzyskanego chromosomu,

FF_sr – średnia wartość funkcji przystosowania najlepszych chromosomów z kolejnych pokoleń.

6.1. Badanie wpływu prawdopodobieństwa mutacji na wyniki

Dla eksperymentów mających na celu sprawdzenie wpływu parametru p_M na wyniki działania algorytmu ustalono następujące wartości parametrów: $n = 6$, $k = 3$, *maxkoszt* = 10, *ifr* = 0.3, $l = 100$, $it = 100$.

Tabela 2 ukazuje skuteczność algorytmu dla różnych wartości p_M. Poszczególne pary kolumn dotyczą znalezienia rozwiązań o określonym koszcie, przy czym pierwsza para kolumn

odnosi się do znalezienia pierwszego rozwiązania akceptowalnego, a ostatnia do znalezienia rozwiązania optymalnego. Wartości w kolumnach *a* wskazują, w którym pokoleniu rozwiązanie o danym koszcie zostało znalezione po raz pierwszy, wartości w kolumnach *b* w ilu na *Ex* eksperymentów nie udało się znaleźć rozwiązania o danym koszcie w *it* pokoleniach. W tabeli umieszczono wartości średnie z 20 eksperymentów.

Tabela 2

Skuteczność algorytmu dla różnych wartości p_M

p_M	Pierwsze rozbicie		sk<31		sk<29		sk<26		sk=23	
	a	b	a	b	a	b	a	b	a	b
0.5	9.30	0	14.65	0	24.75	0	32.18	6	59.33	11
0.4	8.25	0	13.25	0	20.35	0	26.08	4	39.40	9
0.3	6.20	0	10.75	0	15.85	0	28.32	2	40.12	9
0.2	5.20	0	10.35	0	13.60	0	25.05	1	41.08	8
0.1	4.60	0	8.75	0	14.20	0	35.05	1	51.12	4
0.05	4.75	0	8.00	0	13.30	0	26.22	1	41.65	6
0.02	4.30	0	8.88	0	12.93	0	28.00	0	38.77	5
0.01	3.65	0	8.60	0	13.40	0	24.20	0	50.16	7

Tabela 3 przedstawia, jak zmieniają się współczynniki dobroci algorytmu przy zmianie wartości p_M. Wartości współczynników są średnimi z 20 eksperymentów.

Tabela 3

Współczynniki dobroci algorytmu dla różnych wartości p_M

p_M	PzR	RR	V	Rsr	K_min	K_sr	FF_max	FF_sr
0.5	53.10	27.90	0.525	1.52	24.25	32.25	0.0413	0.0316
0.4	67.45	28.60	0.424	1.87	24.85	30.94	0.0403	0.0329
0.3	82.45	23.45	0.284	2.32	24.10	29.30	0.0416	0.0347
0.2	89.40	19.40	0.217	2.99	23.85	28.11	0.0420	0.0360
0.1	92.75	17.40	0.187	3.61	23.50	27.57	0.0426	0.0367
0.05	93.55	11.55	0.123	4.13	23.45	26.42	0.0427	0.0381
0.02	94.00	11.80	0.125	4.25	23.40	26.30	0.0428	0.0384
0.01	94.95	12.40	0.131	4.28	23.70	26.65	0.0423	0.0379

Analizując wyniki zamieszczone w tabeli 2 zauważyć można, że malejąca wartość prawdopodobieństwa zajścia mutacji wpływa na zwiększenie szybkości znajdowania rozwiązań o większym koszcie i obniża procent przypadków, w których w 100 pokoleniach nie udało się znaleźć rozwiązań lepszych.

Z kolei tabela 3 ukazuje, iż malejąca wartość p_M wyraźnie korzystnie wpływa na liczbę pokoleń zawierających rozwiązanie akceptowalne (*PzR*) oraz na średnią liczbę rozbić w każdym pokoleniu (*Rsr*). Zmniejsza się co prawda różnorodność rozwiązań najlepszych (*V*), co świadczy o tym, że algorytm "kręci się" wokół tych samych rozbić zbioru, ale malejący koszt średni (*K_sr*) i minimalny (*K_min*) osiągany w ostatnich pokoleniach i co z tym związane, rosnące wartości

funkcji przystosowania (maksymalne i średnie dla ostatniego pokolenia) świadczą, że są to rozwiązania coraz lepsze.

W czasie badań zauważono, że dla $p_M < 0.02$ wyniki zaczynają się pogarszać. Obserwować to można w tabelach 2 i 3 już dla $p_M = 0.01$. Optymalną wartość prawdopodobieństwa zajścia mutacji najlepiej więc wybrać spośród $p_M > 0.01$.

6.2. Badanie wpływu liczności populacji na szybkość uzyskiwania wyników

Dla eksperymentów mających na celu sprawdzenie wpływu liczności populacji na szybkość znajdowania rozwiązań problemu ustalono następujące wartości parametrów: $n = 6$, $k = 3$, $maxkoszt = 10$, $ifr = 0.3$, $p_M = 0.2$. Liczność populacji l zmieniała się w granicach od 10 do 500 chromosomów, a liczba pokoleń (w zależności od l) it od 500 do 100 pokoleń.

Wyniki w tabelach 4 i 5 przedstawione zostały odpowiednio jak w tabelach 2 i 3 i tak powinny być interpretowane. Prezentowane wartości stanowią również średnią z 20 eksperymentów.

Tabela 4

Skuteczność algorytmu dla różnej liczby chromosomów w populacji

parametry zmienne	Pierwsze rozbicie		sk<31		sk<29		sk<26		sk=23	
	a	b	a	b	a	b	a	b	a	b
$l = 10$ $it = 500$	43.90	0	144.90	0	191.80	0	223.60	10	210.25	12
$l = 20$ $it = 300$	20.55	0	47.25	0	70.30	0	212.75	9	144.60	12
$l = 50$ $it = 200$	7.50	0	18.95	0	28.90	0	74.80	1	92.80	8
$l = 100$ $it = 200$	7.25	0	11.15	0	17.10	0	49.50	1	96.60	3
$l = 200$ $it = 100$	4.20	0	6.40	0	9.30	0	26.10	0	40.00	3
$l = 500$ $it = 100$	3.20	0	4.60	0	5.50	0	8.80	0	27.40	0

Tabela 5

Współczynniki dobroci algorytmu dla różnej liczby chromosomów w populacji

parametry zmienne	PzR	RR	V	Rsr	K_min	K_sr	FF_max	FF_sr
$l = 10$ $it = 500$	15.88	14.8	0.187	1.05	24.80	34.20	0.0405	0.0300
$l = 20$ $it = 300$	40.25	23.45	0.194	1.16	24.75	32.63	0.0406	0.0314
$l = 50$ $it = 200$	68.52	31.20	0.228	1.64	23.85	30.81	0.0420	0.0331
$l = 100$ $it = 200$	84.08	37.00	0.220	2.43	23.35	28.94	0.0429	0.0350
$l = 200$ $it = 100$	91.50	25.95	0.284	3.72	23.50	27.92	0.0426	0.0362
$l = 500$ $it = 100$	96.70	25.70	0.266	6.92	23.00	26.49	0.0435	0.0381

Doświadczenia, których wyniki przedstawiono powyżej, wykazały bezsprzecznie, iż większa liczność populacji wpływa korzystnie na rezultaty poszukiwań rozwiązania. Dla 10 chromosomów w populacji w ciągu 500 pokoleń średnio tylko 15,88% zawierało rozbicia, przy czym w każdym takim pokoleniu występowało średnio $Rsr = 1.05$ rozwiązania akceptowalnego.

Natomiast dla populacji 500-chromosomowej, aż w 96.7% pokoleń osiągnięto takie rozwiązania i było ich w pokoleniu średnio $Rsr = 6.92$. Ponadto analizując wartości średnie i ekstremalne sumy kosztów znalezionych rozbić oraz wartości funkcji przystosowania zauważyć można, iż przy większej liczności populacji osiąga się dużo lepsze wyniki w mniejszej liczbie cykli pokoleniowych. Jeżeli chodzi o czas trwania poszukiwań, to przy 10 chromosomach w populacji przetwarzanie odbywa się wprawdzie szybciej, ale w ponad połowie eksperymentów nie znajduje się rozwiązania optymalnego nawet w 500 pokoleniu, podczas gdy przy populacji liczącej 500 osobników rozwiązanie optymalne osiąga się już w 27 iteracji.

Oczywiście, dobór liczności populacji dla konkretnego problemu określonego przez parametry n i k będzie zależał właśnie od ich wartości, mają one bowiem zasadniczy wpływ na złożoność problemu i moc przestrzeni potencjalnych rozwiązań. Za każdym razem konieczne jest wypośrodkowanie między czasem, w którym chce się osiągnąć rozwiązanie, i możliwością udostępnienia odpowiednio dużego obszaru pamięci a doskonałością rozwiązania.

Przeprowadzone doświadczenia udowodniły jedynie, że łatwiej jest znaleźć dobre rozwiązanie problemu, jeżeli poszukuje się go za pomocą zbioru chromosomów próbkującego przestrzeń rozwiązań dokładniej.

7. Podsumowanie

Spodziewano się, iż zagadnienie rozbicia zbioru ze względu na przynależność do grupy problemów optymalizacji dyskretnej okaże się podatne na zaadaptowanie dla niego mechanizmów genetycznych. Rezultaty są więc zgodnie z oczekiwaniami. Przeprowadzone badania wykazały ponadto, że skuteczność działania systemu bazującego na algorytmie genetycznym zależy w dużej mierze od wartości prawdopodobieństw i zmiennych parametrów, z których korzysta się w algorytmie. Ich optymalne wartości muszą być dobrane empirycznie. W ramach prezentowanej pracy zajmowano się jedynie dwoma z nich. Planowane są dalsze badania mające na celu taki dobór pozostałych parametrów, by system optymalnie wykorzystywał moc, jaką dają mu wzorce zaczerpnięte z natury.

LITERATURA

[1] Cung V.D., Le Cun B.: An Efficient Implementation of Parallel A*. Annals of Artificial Intelligence, INRIA, Avr. 1993.

[2] Goldberg D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company Inc., Reading, Massachusetts 1989.

[3] Holland J.H.: Adaptation in natural and artificial systems. Ann Arbor: The University of Michigan Press, 1975.

[4] Korf R.E.: Optimal Path-finding Algorithms. Search in Artificial Intelligence, red. Kanel I., Kumar V., Springer Verlag, Berlin 1988, s. 223-267.

[5] Kubińska M.: Algorytmy przeszukiwania heurystycznego. Praca dyplomowa-magisterska, Instytut Informatyki Politechniki Śląskiej, Gliwice 1995.

[6] Michalewicz Zb.: Genetic Algorithms + Data Structures = Evolution Programs. II Extended Edition, Springer Verlag, Berlin 1994.

[7] Reingold E.M., Nievergeld J., Deo N.: Algorytmy kombinatoryczne. PWN, Warszawa 1985.

[8] Sysło M.M., Deo N., Kowalik J.S.: Algorytmy optymalizacji dyskretnej. PWN, Warszawa 1993.

[9] Syswerd G.: Schedule Optimization Using Genetic Algorithms. Davis L.: Handbook of genetic algorithms, Pitman, London 1987, s. 332-349.

[10] Whitley D., Starkweather T., Shanner D.: The Travelling Salesman and Sequence Scheduling: Quality Solution Using Genetic Edge Recombination. Davis L.: Handbook of genetic algorithms, Pitman, London 1987, s. 350-372.

Recenzent: Dr Mariusz Boryczka

Wpłynęło do Redakcji 3 grudnia 1996 r.

Abstract

The Set Partitioning Problem (SPP) is a difficult combinatorial optimization problem that belongs to NP-complete problems' class. This means that no algorithm to find an optimal solution with polynomial calculation complexity is known so far. Some attempts to find approximate solutions for this problem with use of various heuristics were taken. Their calculation complexities however remain exponential and demanded amount of operating memory for the searching process in many cases grows exponentially as well.

In this paper a trial of implementation of Genetic Algorithm for SPP is presented. In chapter 2 the Set Partitioning Problem is described. It is explained with use of Delivery Problem that is ilustrated in fig.1. The efficiency of searching process depends on many elements that build GA searching system. The successful application of GAs requires an appropriate encoding of the problem. In chapter 3 the representation of basic genetic algorithm elements which were designed for SPP is presented. The construction of the chromosome that represents the solution is described in section 3.1. Section 3.2. defines an objective function that is constructed of two

components – one that measures the real cost of the solution and the second that penalizes constraint violations – and the fitness function that is defined as a reciprocal of the objective function. The specialized Genetic Algorithm Operators – crossover, mutation and reproduction – have been designed for the problem as well (section 3.3). Chapter 4 describes the algorithm that lets us find the solution of the SPP.

The results of searching process and its efficiency depends on numerous parameters which define the way the algorithm works. The group of GA parameters consists of many elements and their values have to be set individually. In research described in this paper the influence of mutation rate and population size on the quality of the solution has been checked (chapter 6). Numerous experiments have been done and their results are presented in Table 2 and Table 3 – for various values of mutation rate – and Table 4 and Table 5 respectively for the various population sizes .

Conclusions and plans for future research are discussed in chapter 7.