Anna Jasińska-Suwada[*], *Witold Dzwinel*[**], *Krzysztof Rozmus*[**], *Jacek Sołtysiak*[**]

ZASTOSOWANIE METOD NATURALNYCH W PROBLEMACH POSZUKIWANIA OPTYMALNEGO ROZWIĄZANIA

1. Wstęp

Rozwój nauk przyrodniczych oraz potrzeba rozwiązywania złożonych problemów inżynierskich pociągają za sobą coraz większe zapotrzebowanie na moc obliczeniową maszyn cyfrowych. Stało się to przyczyną wzmożonych poszukiwań zarówno w sferze sprzętowej, jak i programowej. Wzrosło zainteresowanie obliczeniami rozproszonymi i pojawiły się środowiska programowania do ich realizacji [33]. Możliwość przetwarzania współbieżnego pociągnęła za sobą rozwój nowych, lepiej dostosowanych do równoległych obliczeń technik. Wśród nich można wyróżnić grupę określaną mianem: „natural solvers" [11]. Inspiracją do ich powstania była obserwacja zdarzeń zachodzących w przyrodzie: procesu ewolucji, zachowania różnych gatunków zwierząt i procesów fizycznych, oraz zachwyt nad skutecznością, a zarazem prostotą tych zjawisk. Niektóre z tych technik, będących efektem naśladowania natury, charakteryzują się inherentną równoległością, która umożliwia ich efektywną implementację na współczesnych wieloprocesorowych systemach komputerowych. Narzędzia te, będące alternatywą do podejścia klasycznego, dają prosty model komputerowy, wolny od ograniczeń modelu funkcyjnego. Tak jak w przyrodzie, złożone i skomplikowane procesy i struktury opisywane są za pomocą dużej ilości wzajemnie oddziałujących, prostych obiektów. Metody naturalne służą do modelowania zjawisk fizycznych i przeprowadzania symulacji komputerowej. Ich skuteczne implementacje, pozwalające rozwiązywać wiele problemów nauki i techniki, świadczą o tym, że świat ludzkiego umysłu poddaje się takim samym prawom jak natura, że istnieją uniwersalne heurystyki. Pozostaje pytanie, czy „odkrywanie" tych heurystyk nie jest także elementem naturalnego procesu ewolucyjnego – odpowiedzią na wyzwania nauki.

[*] Instytut Modelowania Komputerowego, Politechnika Krakowska
[**] Katedra Informatyki, Akademia Górniczo-Hutnicza, Kraków

Wiele problemów naukowych, inżynierskich czy ekonomicznych sprowadza się do poszukiwania najlepszego rozwiązania lub minimalizacji (maksymalizacji) pewnej wielowymiarowej funkcji celu [25, 31]. Optymalizacja statyczna (programowanie matematyczne) zajmuje się wyznaczeniem wektora, należącego do zbioru dopuszczalnych rozwiązań, dla którego funkcja celu osiąga ekstremum. W zależności od charakteru funkcji celu oraz ograniczeń można wyróżnić programowanie liniowe i nieliniowe. Rozwiązywaniem zadań optymalizujących procesy przebiegające w pewnym czasie, polegających na odnalezieniu sterowania minimalizującego lub maksymalizującego wskaźnik jakości, przy spełnieniu narzuconych ograniczeń zajmuje się optymalizacja dynamiczna. Jeśli czas występuje w postaci dyskretnej, można zastosować metodę programowania dynamicznego, opartą na zasadzie Bellmana [1]. Rekurencyjna zależność zawarta w niej umożliwia redukcję dużego problemu obliczeniowego do szeregu mniejszych, kolejno rozwiązywanych. Jednak z większymi problemami obliczeniowymi, dotyczącymi poszukiwań najlepszego rozwiązania dla zadań o dużej liczbie minimów lokalnych, czy problemów kombinatorycznych, klasyczne metody optymalizacji sobie nie radzą.

Metody poszukiwania optimum wielowymiarowej i wielomodalnej funkcji można podzielić na: analityczne, enumeratywne i losowe [2]. Wśród metod analitycznych wyróżnia się metody pośrednie, gdzie szukanie ekstremów sprowadza się do rozwiązania układu równań, wynikającego z przyrównania gradientu funkcji do zera, oraz bezpośrednie, polegające na poruszaniu się wzdłuż wykresu funkcji, w kierunku wyznaczonym np. przez gradient funkcji. Metody analityczne cechują się brakiem odporności ze względu na lokalny zakres poszukiwań oraz warunki istnienia pochodnych. Rozwiązywaniem zadań nieliniowych z ograniczeniami równościowymi i nierównościowymi zajmuje się teoria Kuhna i Tuckera [45]. Wykazali oni, że warunkiem koniecznym istnienia ekstremum warunkowego funkcji wielu zmiennych w punkcie jest spełnienie przez ten punkt warunków koniecznych punktu siodłowego funkcji Legrange'a. Dla odnalezienia globalnego ekstremum konieczne jest przeglądnięcie wartości badanej funkcji we wszystkich punktach spełniających ten warunek. Algorytmy enumeratywne polegają na przeglądaniu obszaru poszukiwań i odnalezieniu rozwiązania optymalnego. Są więc proste, skuteczne, nieczułe na złożoność funkcji, ale zupełnie nieefektywne. Metody poszukiwania losowego (np. błądzenie przypadkowe) polegają na przeglądaniu losowo wybranych elementów i są również nieefektywne.

Optymalizacja jest naturalną cechą wszystkich procesów zachodzących w przyrodzie i tę własność wykorzystują naturalne metody. Naturalne techniki zostały zastosowane w heurystycznych metodach poszukiwania najlepszego rozwiązania. Ich wewnętrzna, wbudowana współbieżność sprawia, że implementacja tych metod, dostosowana do równoległych architektur sprzętowych nie przedstawia dużego problemu. Do najważniejszych przykładów naturalnych metod znajdywania minimum trzeba zaliczyć:

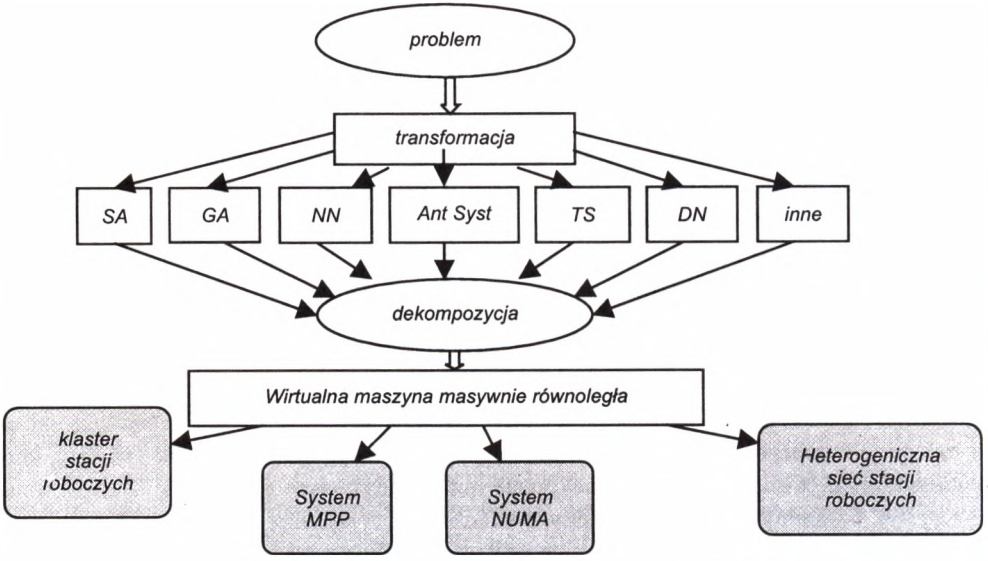
- SA – Symulowane odprężanie (Simulated Annealing) – będące techniką optymalizacji, która naśladuje fizyczny proces oziębiania materiałów. Wykorzystuje się tu podstawowe prawo fizyki statystycznej i tendencję układów fizycznych do minimalizacji własnej energii potencjalnej [3, 4, 12],
- GA – Algorytmy genetyczne – inspirowane procesem ewolucji w przyrodzie: mechanizmami doboru naturalnego i dziedziczności [2, 26, 34, 35, 36, 38],
- MD – Dynamika molekularna (metoda cząstek) – realizująca model dynamiki wzajemnie oddziałujących cząstek [5, 6],

- SNN – Sztuczne sieci neuronowe – wykorzystujące matematyczny, uproszczony model komórki nerwowej i sposób uczenia zespołów złożonych z takich elementów [7, 8],
- Algorytm mrówkowy (Ant System) – wykorzystujący model kolonii „sztucznych mrówek", jako narzędzia optymalizacji [9],
- TS – Metoda tabu search – gdzie tymczasowe rozwiązanie porusza się po trajektorii, w przestrzeni rozwiązań, dokonując w każdym kroku wyboru najlepszego z dozwolonych elementarnych ruchów [10, 3].

Heurystyki naturalne stały się obok metod klasycznych najważniejszym narzędziem poszukiwania najlepszego rozwiązania. Głównym kryterium oceny metody jest odporność, będąca kompromisem pomiędzy efektywnością a skutecznością [2]. Metody analityczne cechują się najwyższą efektywnością, lecz w wąskim obszarze zastosowań – dla funkcji jednomodalnych. W przypadku problemów kombinatorycznych, czy wielomodalnych są zupełnie nieskuteczne [2]. Natomiast metody oparte na heurystykach naturalnych są ogólne i można je z dobrym skutkiem stosować we wszystkich typach problemów [16, 23], choć dla niektórych z nich będą mniej efektywne niż metody wyspecjalizowane. Zastosowanie heurystyk naturalnych może być pierwszym krokiem w rozwiązywaniu problemów optymalizacji. Po przeprowadzonej przez nie wstępnej analizie można zastosować lokalną, wyspecjalizowaną metodę [46]. Taka hybrydyzacja umożliwia skonstruowanie metody łączącej zalety różnych algorytmów.

Rysunek 1 przedstawia model transformacji (mapowania z ang. mapping) problemu obliczeniowego, polegającego na poszukiwaniu najlepszego rozwiązania, na masywnie równoległy system komputerowy. Proces transformacji należałoby podzielić na kilka etapów [11, 42, 43, 44]. W pierwszym kroku mamy do czynienia z odwzorowaniem sformułowanego zadania na wybraną metodę. Etap ten jest procesem twórczym i na razie nie mógłby być dokonywany automatycznie. Następnym krokiem jest dekompozycja problemu, z wykorzystaniem wewnętrznej równoległości metody. Ze względu na skończoną ilość metod i dużą liczbę prac dotyczących ich efektywnego zrównoleglenia etap ten można by było traktować jako częściowo ukryty dla badacza. Mógłby on jednak ingerować w strukturę algorytmu, by dopasować go jak najlepiej do rozwiązywanego problemu. Tak podzielone zadanie może być transformowane na wirtualną, równoległą maszynę i implementowane na różnych równoległych systemach komputerowych. Ten etap w przyszłości mógłby być całkowicie przeźroczysty dla użytkownika, który nie musiałby ingerować w sposób implementacji i zrównoleglenia wybranej metody na danym systemie.

Niniejsza praca jest pierwszą częścią serii artykułów poświęconych heurystycznym algorytmom optymalizacyjnym, opartych na metodach naturalnych. Celem artykułu jest krótkie przedstawienie najważniejszych z nich oraz szczegółowa analiza algorytmów genetycznych i metody cząstek. Dla tych ostatnich zbadane zostaną sposoby przeszukiwania przestrzeni rozwiązań, heurystyki optymalizacyjne oraz sposób doboru parametrów. Następnie opisane będą zasady porównywania algorytmów oraz wyniki testów, mających na celu porównanie algorytmu genetycznego i uproszczonej metody cząstek. Badania przeprowadzone zostały przy użyciu testowych problemów poszukiwania minimów wielowymiarowych i wielomodalnych funkcji. Na końcu zebrane zostaną wnioski i przedyskutowane najważniejsze rezultaty wykonanej pracy.



Rys. 1. Model transformacji problemu obliczeniowego na równoległy system komputerowy

2. Przegląd metod naturalnych

2.1. Symulowane odprężanie

Przykładem stochastycznej metody Monte-Carlo jest algorytm Metropolisa [5, 12], oparty na doświadczeniach, związanych z oziębianiem materiałów. Powolne ochładzanie może doprowadzić do uzyskania struktury kryształu, o minimalnej wartości energetycznej. Zgodnie z zasadami mechaniki statystycznej, układ zmierzając do stanu minimalnej energii, poprzez jej fluktuację, zależną od temperatury, może zwiększać swoją energię o ΔE_p z prawdopodobieństwem p, określonym wzorem

$$p = e^{\frac{-\Delta E_p}{kT}} \tag{1}$$

gdzie:

T – temperatura układu,
k – stała Boltzmanna.

Przy wyższych temperaturach, często zdarzają się zmiany aktualnego stanu, co odpowiada przeszukiwaniu dużego zakresu rozwiązań. Wraz z oziębianiem układu, przejścia do stanu o wyższej energii są coraz rzadsze i rozwiązanie stabilizuje się. Akceptacja zmian, które przynoszą „gorsze rozwiązanie", umożliwia algorytmowi wyjście z lokalnych minimów. Dzięki tym własnościom schemat Metropolisa został wykorzystany w poszukiwaniu minimum globalnego wielowymiarowych funkcji kryterium przez Kirkpatricka [13]. Ze względ-

du na sposób przeszukiwania, obejmujący duży obszar rozwiązań, jest on szczególnie efektywny w rozwiązywaniu zadań kombinatorycznych. Na nim opiera się działanie maszyn Boltzmanna [7].

Schemat obliczeniowy metody Metropolisa przedstawia rysunek 2.

```
procedure SM;{
i = początkowe rozwiązanie;
T = T0;     (wysoka wartość „temperatury początkowej")
oblicz Ei;  (wartość „energii"));
repeat
   {
     wygeneruj nowe rozwiązanie j (operator sąsiedztwa)
     oblicz Ej;
     if (Ej < Ei)
       then
            i = j;
     else
       if (random[0,1) < exp (-(Ej - Ei)/T))
            then  i = j;
     T = T - ΔT;  }
 until warunek stopu;}
```

Rys. 2. Schemat Metropolisa

Schemat Metropolisa, będący algorytmem symulowanego odprężania – SA (*Simulated Annealing*), stał się podstawą dla wielu metod poszukiwania najlepszego rozwiązania. Różnią się one przyjętym kryterium akceptacji nowego stanu. W metodzie Duecka i Scheuera [5, 14], zamiast obliczania prawdopodobieństwa, danego wzorem (1), zastosowano kryterium malejącego progu. Jeśli zmiana „energii" (wartości funkcji celu) przekracza wartość progu, rozwiązanie jest odrzucane. Inne metody bazujące na SA [5, 15, 4] to: SQ (*Simulated Quenching*), ST (*Simulated Tempering*), FA (*Fast Annealing*), MFA (*Mean Field Annealing*) – wykorzystujące zmodyfikowany model odprężania, ASA (*Adaptive Simulated Annealing*) Listera Ingbera, początkowo znana jako (VFSR – *Very Fast Simulated Reannealing*), wykorzystująca adaptcję i śledzenie zbieżności w każdym wymiarze *n*-wymiarowej przestrzeni [15, 42, 41] oraz quantum-mechanical annealing [16], bazujące na równaniu Schrödingera.

2.2. Algorytm genetyczny

Termin EC (*Evolutionary Computation*) [17] – obliczenia ewolucyjne obejmuje wiele technik obliczeniowych, w których kluczowym elementem jest model procesów ewolucyjnych. Ewolucyjne algorytmy operują na populacji struktur, które ewoluują, wykorzystując reguły selekcji oraz operatory mutacji i rekombinacji. Każdy element populacji oceniany jest za pomocą miary przystosowania do środowiska (*fitness factor*).

Wśród algorytmów ewolucyjnych wyróżnia się [17]:

- programowanie ewolucyjne (EP),
- strategie ewolucyjne (ES),
- algorytmy genetyczne (GA).

Typowy algorytm ewolucyjny przedstawiony jest na rysunku 3.

```
procedure EA;{
t = 0;
inicjuj populację P(t);
oceń P(t);
repeat {
                t = t + 1;
        wybór rodziców P(t);
        rekombinacja P(t);
        mutacja P(t);
        oceń P(t);
        selekcja P(t);}
until warunek stopu;}
```

Rys. 3. Algorytm ewolucyjny

Podstawowymi elementami wszystkich algorytmów ewolucyjnych jest populacja i działające na niej operatory. Różnice pomiędzy algorytmami tkwią w wyborze:

- reprezentacji indywidualnych struktur,
- typu mechanizmów selekcji,
- rodzaju genetycznych operatorów,
- metody ich oceny.

W programowaniu ewolucyjnym (EP) nie stosuje się operacji krzyżowania, a indywidualne struktury zadawane są w postaci wektorów rzeczywistych, list lub grafów. Wybór rodziców nie zależy od funkcji przystosowania, a selekcja jest dokonywana na podstawie probabilistycznej funkcji, bazującej na przystosowaniu.

Ewolucyjne strategie (ES) stosują reprezentację wektorów rzeczywistych. Rodzice wybierani są losowo, stosuje się mutację, a następnie rekombinację, która tu jednak ma drugorzędne znaczenie. Operator selekcji wybiera najlepiej przystosowane indywidualne struktury.

Algorytmy genetyczne używają zwykle niezależnej reprezentacji w postaci ciągu bitów, choć spotyka się też inne reprezentacje (grafy, wyrażenia Lispu, listy, wektory) [17]. Selekcja rodziców zależna jest od wartości funkcji przystosowania. Dzieci utworzone w wyniku rekombinacji rodziców poddawane są mutacji i krzyżowaniu. Relacje między tymi operatorami są przeciwne niż w ES. Mutacja, polegająca na inwersji bitów z małym prawdopodobieństwem, ma znaczenie drugorzędne. Zastosowanie algorytmu genetycznego do rozwiązania problemu wymaga określenia następujących elementów:

- sposobu reprezentacji poszczególnych osobników populacji (tj. możliwych rozwiązań),
- sposobu oceny elementów populacji,
- operatorów genetycznych.

Łańcuchy zer i jedynek stosowane w algorytmach genetycznych są najprostszą reprezentacją, niezależną od problemu. Można przy jej pomocy budować skomplikowane struktury, co wymaga jednak kodowania. Każde rozwiązanie problemu musi mieć swoją reprezentację w postaci łańcucha i każdy łańcuch musi reprezentować jakieś rozwiązanie. Elementy populacji oceniane są za pomocą funkcji dopasowania, reprezentującej rozwiązywany problem. Musi być ona zdefiniowana w całej przestrzeni łańcuchów.

Operator selekcji, wielkość populacji oraz generacja populacji początkowej mają wpływ na różnorodność populacji. Jest ona bardzo ważnym aspektem algorytmów genetycznych, gdyż decyduje o sposobie przeszukiwania dziedziny i od niej zależy możliwość znalezienia najlepszego rozwiązania. Na początku algorytm sprawdza wiele różnych rozwiązań, a następnie dokładniej penetruje potencjalnie lepsze obszary. Jeśli początkowa populacja jest zróżnicowana, zawiera całościowy przekrój dziedziny algorytmu i algorytm ma szansę na odnalezienie najlepszego rozwiązania. Źle przygotowana populacja startowa może sprawić, że pewne obszary dziedziny nie będą w ogóle badane.

Algorytm genetyczny nie gwarantuje zbieżności do minimum globalnego, dlatego stosuje się hybrydy. Połączenie AG z SA zaowocowało powstaniem algorytmu PRASA (*Parallel Recombinative Simulated Annealing*) [12].

Schemat metody przedstawia rysunek 4.

```
procedure PRASA;{
T = T0;      ustal wysoką początkową temperaturę
Inicjuj n-elementową populację;
repeat {
do n/2 razy{
                t = t + 1;
      wybór 2 rodziców z populacji;
      generuj 2 potomków (mutacja, rekombinacja);
      oceń potomków (Ej);
      przeprowadź zawody Boltzmanna;
      zastąp rodziców zwycięzcami zawodów;}
 T = T – ΔT;  }
until warunek stopu;}
```

Rys. 4. Algorytm PRASA

Odpowiada on równolegle działającym wielu kopiom SA, z wykorzystaniem operatorów genetycznych: mutacji i krzyżowania. Zastosowanie schematu Metropolisa gwarantuje zbieżność metody do minimum globalnego [12]. Tradycyjny operator selekcji zastąpiono tzw. zawodami Boltzmanna, w których prawo rodziców do życia określone jest prawdopodobieństwem p, danym wzorem

$$p = \frac{1}{1 + e^{(E_i - E_j)/T}} \tag{2}$$

gdzie:

E_i – „energia" rodzica,

E_j – „energia" potomka.

2.3. Dynamika molekularna – metoda cząstek

Metody takie jak: SA, GA, PRASA, TS, sieci Hopfielda, algorytm mrówkowy nadają się do rozwiązywania problemów kombinatorycznych, z dyskretną przestrzenią rozwiązań. Przy poszukiwaniu minimum globalnego wielowymiarowych funkcji ciągłych wydaje się, iż bardziej efektywny jest algorytm bazujący na dynamice molekularnej.

Dynamika molekularna (MD – *Molecular Dynamics*) [6, 5] jest techniką obliczeniową stosowaną od wielu lat w problemach symulacji N wzajemnie oddziałujących ciał. Podstawowymi obiektami w MD są cząstki, których zachowanie wzorowane jest na dynamice molekuł realnego świata. Każda cząstka oddziałuje z innymi – znajdującymi się w pewnym sąsiedztwie, ograniczonym promieniem kuli R_{cut} – siłą zależną od gradientu energii potencjalnej. Przykładem potencjału dwucząstkowego stosowanego w dynamice molekularnej jest krótkozasięgowy potencjał Lennarda-Jonesa [5]. Zgodnie z nim cząstki przyciągają się, a w pewnej odległości zaczynają się odpychać. W każdym kroku algorytmu obliczana jest całkowita siła działająca na każdą z cząstek, będąca sumą sił pochodzących od wszystkich cząstek znajdujących się w otoczeniu. Korzystając z równań ruchu Newtona, wyznaczane są także prędkości i położenia cząstek. Warunkiem poprawności modelu jest stałość całkowitej energii układu. Opisana metoda znalazła zastosowanie w badaniu fizycznych właściwości gazów, cieczy i ciał stałych oraz w symulacji zjawisk i procesów w mikroskali. Została także wykorzystana do poszukiwania globalnego minimum funkcji wielowymiarowych [5, 6].

O ile w przyrodzie molekuły dążą do osiągnięcia stanu minimalnej energii, o tyle wirtualne molekuły mogą być wykorzystane do poszukiwania minimum funkcji. Cząstki znajdują się w polu, wyznaczonym przez minimalizowaną funkcję. Oddziałując między sobą i reagując na działanie pola, zmieniają swoje atrybuty – położenia i prędkości. Na poruszające się cząstki działa dodatkowo siła tarcia, dyssypująca energię. Symulacja prowadzi do stanu, w którym cząstki osiągają stabilne położenie i niektóre z nich wskazują na szukane minimum globalne funkcji.

Przyjęto następujące założenia [5]:

1) $F(X) \in C^1$, $X = (x_1, x_2, \ldots, x_n)$,
$X \in \Re^N \wedge \exists X^G \in \Re^N; \forall X \in \Re^N \;\; F(X^G) \le F(X)$;
2) $\{X_1, X_2, \ldots, X_i, \ldots, X_m\}$ jest zbiorem cząstek, $X_i \in \Re^N$;
3) cząstki oddziałują ze sobą siłami f_{oddz}, gdy znajdą się w odległości mniejszej niż R_{cut} oraz odbijają się od granic obszaru zmienności funkcji;
4) $E_p = a \cdot F(X)$ – energia potencjalna, $E_K = \frac{1}{2} \cdot \sum_i m_i \cdot V_i^2$ – energia kinetyczna,
gdzie: $V_i = \dfrac{\mathrm{d}}{\mathrm{d}t} X_i$, t – czas;
5) $F_{tarcia} = -\lambda \cdot V_i$ – siła tarcia, dyssypująca energię, działającą na cząstkę i, W – całkowita praca sił dyssypacyjnych,
6) całkowita energia zbioru cząstek $E_T = E_P + E_K + W = \text{const}$.

Układ równań ruchu Newtona w przestrzeni $\Re^N$ wyraża się wzorem:

$$m_i \cdot \frac{\mathrm{d}V_i}{\mathrm{d}t} = -a \cdot \nabla F(X_i) - \lambda \cdot V_i + k \cdot f_{oddz} \qquad (3)$$

$$\frac{\mathrm{d}X_i}{\mathrm{d}t} = V_i .$$

Gradient energii potencjalnej $-a \cdot \nabla F(X)$, reprezentuje siłę F_i działającą na cząstkę i. Dla poprawnie dobranych parametrów m_i, a, k, λ, dt rozwiązaniem tego układu równań jest stan, w którym cząstki osiągają stabilne położenia w minimach funkcji $F(X)$ i $E_K = 0$. Położenia niektórych z nich wskazują na minimum globalne.

Na ruch cząstek w MD mają wpływ dwa czynniki:

1) siły oddziaływań międzycząstkowych, sprawiające, że przeszukiwana jest cała dziedzina;
2) pole, pochodzące od minimalizowanej funkcji, które skierowuje cząstki w kierunku szukanego minimum.

W wyższej temperaturze, gdy ruch cząstek jest wzmożony, dynamika molekularna przypomina algorytm symulowanego wyżarzania, natomiast wraz ze zmniejszaniem prędkości cząstek, czyli dla niskiej temperatury układu, metoda staje się coraz bardziej podobna do metod gradientowych.

Algorytm metody cząstek przedstawiony jest na rysunku 5.

```
procedure MD;{
inicjuj zmienne: t = 0, R_max, położenia i prędkości m cząstek;
repeat  {
for i = 1 to m do
        for j = 1 to m do
            oblicz wart. funkcji oddział. między cząstkami i i j
for i = 1 to m do
            oblicz wartość siły działającej na cząstkę i;
            oblicz prędkość cząstki i oraz nowe położenie cząstki i;
t = t + 1 }
until warunek stopu;}
```

Rys. 5. Algorytm MD

2.4. Inne metody heurystyczne

Tabu search

TS – *Tabu Search* [10, 3] reprezentuje heurystykę, stosowaną dla kombinatorycznych zadań optymalizacyjnych. Bazuje na operatorze sąsiedztwa i pamięciowych mechanizmach, zabezpieczających przed zapętleniem poszukiwań. Punkt reprezentujący rozwiązanie porusza się w przestrzeni, poczynając od pozycji startowej, za pomocą elementarnych przesunięć. W każdym kroku wybierany jest taki ruch, który gwarantuje najkorzystniejszą wartość funkcji kryterialnej. Dopuszczalne są zmiany, które zwiększają koszty – one umożliwiają wyjście z lokalnych minimów. Zabronione jest cofanie się, aby nie dopuścić do zapętlenia algorytmu. Dlatego określa się zbiór ruchów zakazanych – *tabu* (stad nazwa metody), który jest modyfikowany i ma formę kolejki typu FIFO o pewnej ustalonej długości T. W każdym kroku dodawane jest do niej przesunięcie odwrotne do ostatnio wykonanego, a zakaz wprowadzony T iteracji temu jest usunięty.

Niech:

F – zbiór punktów;

E – minimalizowana funkcja kosztu $E: F \rightarrow \Re$ [1];

$N(f)$ – operator sąsiedztwa punktu f;

M – zbiór elementarnych przesunięć;

$N(f) = \{g \in F: g = \mu(f), \mu \in M\}$.

Jeśli F jest zbiorem binarnych ciągów o skończonej długości L, operator μ może polegać na zmianie (zanegowaniu) jednego z bitów ciągu, co odpowiada genetycznemu operatorowi mutacji. W każdej iteracji t, zbiór elementarnych ruchów podzielony jest na dopuszczalne $A^{(t)}$ i zabronione $T^{(t)}$. Dla startowego stanu $f^{(0)}$: $A^{(0)} = M$, $T^{(0)} = \varnothing$.

W iteracji t dla aktualnego punktu $f^{(t)}$ wybierany jest najlepszy ruch ze zbioru $A^{(t)}$

$$f^{(t+1)} = \mu^{(t)}(f^{(t)}), \text{ gdzie } \mu^{(t)} = \arg\min_{\upsilon \in A(t)} E(\upsilon(f^{(t)})).$$

Ten dyskretny, dynamiczny proces tworzy zbiór punktów, nazywany trajektorią. Zbiór akceptowalnych ruchów nie może być pusty. Obecność zbioru tabu ma zabezpieczać przed wejściem algorytmu w nieskończoną pętlę. Gdy punkt znajdzie się w minimum lokalnym: $E(f^{(t+1)}) = E(\mu^{(t)}(f^{(t)})) > E(f^{(t)})$, wówczas jest możliwe, że następny krok będzie odwrotny do przedniego: $\mu^{(t+1)} = (\mu^{(t)})^{-1}$, co prowadzi do zapętlenia. Dlatego w każdym kroku do zbioru $A^{(t)}$ dodawany jest elementarny ruch, będący inwersją ostatniego. Taki zakaz musi być usuwany po pewnej liczbie iteracji T, gdyż mógłby uniemożliwić uzyskanie optimum w późniejszych fazach. Zbiory: tabu i dopuszczalnych ruchów modyfikowane są w każdym kroku w następujący sposób:

$$T^{(t+1)} \leftarrow T^{(t)} \cup \{(\mu^{(t)})^{-1}\} \setminus \{(\mu^{(\tau)})^{-1}, \text{ dla } \tau : \tau \Leftarrow (t - T)\}$$

$$A^{(t+1)} \leftarrow A^{(t)} \cup \{(\mu^{(t)})^{-1}\} \setminus \{(\mu^{(\tau)})^{-1}, \text{ dla } \tau : \tau \Leftarrow (t - T)\}$$

Algorytm TS przedstawia rysunek 6.

```
procedure TS;{
f(0) = wygenerowana konfiguracja startowa;
t = 0;
A(0) = M;
T(0) = ∅;
repeat
    { μ = najlepszy ruch z dopuszczalnych;
      f(t+1) = μ f(t);
      modyfikuj zbiory T, A;
      t = t + 1;
      if E(f(t)) < E_b then
          { E_b = E(f(t));
            f_b = f(t);  }}
 until warunek stopu;}
```

Rys. 6. Algorytm Tabu Search

Prawidłowy dobór liczby iteracji T, po których zakaz ze zbioru tabu jest usuwany, umożliwiający rozwiązanie zadania, a zarazem nie dopuszczający do zapętlenia jest kłopotliwy.

Dlatego modyfikacja algorytmu TS – RTS (*Reactive Tabu Search*) [10] wprowadza dodatkowo mechanizm, zmieniający wartość T tak, aby była ona zgodna z lokalną strukturą problemu. Początkowo $T^{(0)}$ przyjmuje wartość 1, a następnie, w reakcji na występujące powtórzenia jest modyfikowana. Metoda ta wymaga pamiętania dodatkowych wartości, związanych z wykonanymi ruchami (numer iteracji, w której przesunięcie wykonano) i osiągniętymi stanami (numer iteracji, w której stan został osiągnięty oraz liczba jego powtórzeń). Jeśli algorytmowi grozi zapętlenie (zbyt duża ilość konfiguracji przekroczyła limit powtórzeń), następuje sekwencja losowych ruchów, umożliwiająca wyjście z krytycznego obszaru. Ważną cechą opisanej metody, wyróżniającą ją wśród innych heurystyk, jest fakt, że w dużym stopniu wykorzystuje ona historię wykonanych kroków i w związku z tym wymaga dodatkowych zasobów pamięci.

Algorytm mrówkowy

Jak pokazują badania nad społecznością mrówek, na trasie swoich przejść pozostawiają one „feromonowy szlak", którego intensywność jest drogowskazem dla kolejnych osobników. Proces ten charakteryzuje się dodatnim sprzężeniem zwrotnym. Prawdopodobieństwo wybrania ścieżki przez mrówkę jest tym większe, im więcej mrówek wybrało poprzednio tę trasę. Sposób, w jaki mrówki odnajdują najkrótszą drogę między mrowiskiem i żerowiskiem, stał się inspiracją dla twórców algorytmu. Znajduje on zastosowanie głównie w rozwiązywaniu problemów kombinatorycznych, dlatego zostanie przedstawiony na przykładzie problemu komiwojażera dla n miast.

Algorytm mrówkowy, opisany przez Dorigo [9] przedstawiony jest na rysunku 7.

```
procedure AM;{
ustal wartości początkowe (il_cykli = 0, τ_ij(t) = c, Δτ_ij = 0;
umieść m mrówek w n węzłach;
repeat
    dla każdej mrówki dodaj startową pozycję do listy tabu;
    for i = 1 to n – 1 do {
        for k = 1 to m do {;
            wybierz miasto j z prawdopodobieństwem p zależnym od
            oddalenia i intensywności szlaku;
            przesuń mrówkę k-tą do j;
            dodaj j do listy tabu mrówki m; } }
    for k = 1 to m do {
        oblicz całkowita długość drogi L_k dla mrówki k;
        if (L_k < L_min) then L_min = L_k;
        mrówkę k przenieś do punktu startowego};
    for i = 1 to n do {
        for j = 1 to n do {
            policz intensywność feromonową τ_ij(t + n);
            Δτ_ij = 0; } }
     t = t + n
     il_cykli = il_cykli + 1;
     for k = 1 to m do {
         wyczyść listę tabu dla mrówki k; } }
 until (il_cykli >= il_max);
 wypisz najkrótszą drogę L_min;}
```

Rys. 7. Algorytm mrówkowy

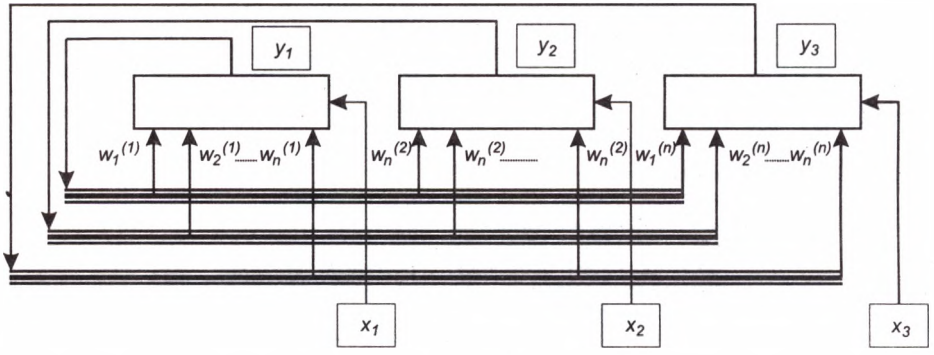
Zadaniem każdego osobnika (jest ich m) jest odwiedzenie wszystkich miast. Mrówki są prostymi agentami, które wykorzystują swoją wiedzę i doświadczenia innych. Początkowo zostają rozmieszczone w n węzłach (miastach). Kierują się następującymi zasadami:

- wybór kolejnego miasta na ich trasie jest zależny od jego oddalenia oraz intensywności szlaku pozostawionego przez inne mrówki,
- mrówka może wybrać tylko takie miasta, w których jeszcze nie była, co jest sterowane zakazaną listą (*tabu* list),
- po zakończeniu całej trasy pozostawia pewną ilość feromonu na każdej z gałęzi przebytego szlaku.

W każdej iteracji, każda mrówka odwiedza kolejne miasto. Po pełnym cyklu algorytmu, modyfikowana jest intensywność szlaku na każdej gałęzi.

SSN Sieci Hopfielda

Sieć Hopfielda [7] jest przykładem sieci o strukturze dwukierunkowej (sieć rezonansowa), której działanie oparte jest na samorzutnej minimalizacji funkcji energii. Wykorzystuje ona sprzężenie zwrotne – wszystkie neurony połączone są ze sobą w ten sposób, że wyjście każdego z nich jest równocześnie wejściem każdego, jak pokazuje rysunek 8.



Rys. 8. Schemat sieci Hopfielda

Pracę sieci można zinterpretować w następujący sposób: początkowo wszystkie neurony pobudzone są na wejściu sygnałami, w wyniku czego na wszystkich wyjściach pojawi się również sygnał. W każdym bloku m, w czasie j realizowane jest sumowanie sygnałów wg wzoru

$$e_m^{(j)} = \sum_{i=1,n} w_i^{(m)} y_i^{(j)} + x_m^{(j)} \tag{4}$$

Jeśli suma ta przekracza wartość progową $w_0^{(m)}$ na wyjściu generowany jest sygnał równy 1, w przeciwnym wypadku –1. Następnie sygnał wejściowy zanika, natomiast w sieci przebiega dynamiczny proces generowania kolejnych stanów wyjściowych neuronów. W zależności od współczynników wagowych $w_i^{(m)}$, łączących wyjście i-tego neuronu z wejściem

m-tego, proces ten może być oscylacyjny, rozbieżny lub zbieżny do pewnej wartości Y^* (wektor sygnałów wyjściowych neuronów), co odpowiada zatrzymaniu procesu. Wykazano, że sieć asocjacyjna symetryczna ($w_i^{(m)} = w_m^{(i)}$), bez autoasocjacji pojedynczych neuronów ($w_m^{(m)} = 0$), generuje stabilne rozwiązanie. Rozwiązanie to odpowiada osiągnięciu przez układ stanu o minimalnej „energii". Wybór właściwego stanu docelowego sieci jest równoznaczny z globalnym minimum funkcji „energetycznej". „Energia" jest pojęciem umownym i odpowiada funkcji Lapunowa dla układów dynamicznych oraz funkcji celu w optymalizacji. Model sieci Hopfielda odpowiada fizycznemu modelowi magnetyków i szkieł spinowych Isinga [8]. Spin skierowany „w górę" odpowiada aktywacji neuronu, a zwrócony „ku dołowi" jest równoważny stanowi spoczynku. Na każdy spin oddziałuje pole magnetyczne, określające jego dynamikę. Jego źródłem są pozostałe spiny oraz pole zewnętrzne. Widać tu wyraźną analogię do funkcji aktywacji neuronu. Wiedza dotycząca modeli Isinga może zostać wykorzystana do analizy działania sieci neuronowych.

Za uogólnienie sieci Hopfielda uważana jest Maszyna Boltzmanna [7]. Jest to sieć stochastyczna, w której stan układu (sygnały wyjściowe wszystkich neuronów) zmienia się w sposób losowy, z prawdopodobieństwem określonym przez rozkład Boltzmanna z mechaniki statystycznej. Podstawę działania maszyny Boltzmanna jest algorytm symulowanego odprężania. Aktywacja neuronu m, w kroku j dokonuje się z prawdopodobieństwem określonym wzorem

$$p_m^{(i)} = 1/\left[1 + \exp\frac{-\delta \cdot E_m^{(j)}}{T^{(j)}}\right] \tag{5}$$

gdzie:

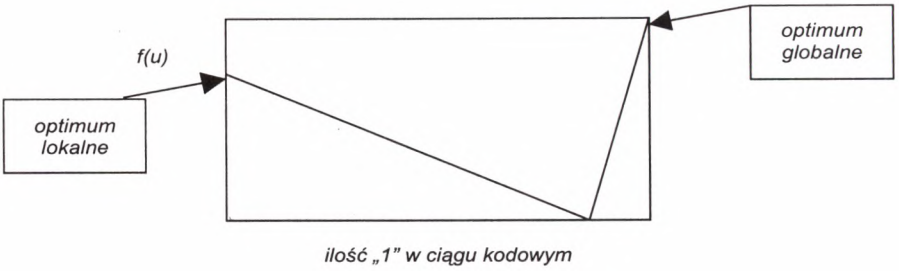
δ – stała,
$T^{(j)}$ – „temperatura" sieci w kroku j,
$E_m^{(j)}$ – nadwyżka pobudzenia m-tego neuronu (ponad wartość progową).

Obniżając temperaturę, proces zmierza do stanu równowagi.

3. Sposoby porównywania metod poszukiwania optimum

3.1. Problemy trudne

Zagadnienia sprawiające wiele problemów programom optymalizacyjnym cechują się wielomodalnością, są to często problemy zwodnicze, a także zagadnienie długiej ścieżki. Problemy zwodnicze [22, 21] są to zadania, które wprowadzają w błąd algorytm, szukający optymalnego rozwiązania, sprawiając, że odnalezione zostanie optimum lokalne. Podstawę ich konstrukcji stanowią funkcje pułapkowe (*trap functions*), przy pomocy których można zbudować wielomodalne funkcje zwodnicze o milionach optimów lokalnych. Charakter zwodniczej funkcji pułapkowej [22, 18] przedstawia rysunek 9. Cechuje się ona obecnością optimów, z których jedno – lokalne otoczone jest dużą ilością osobników o wysokiej wartości funkcji przystosowania, natomiast drugie – globalne jest silnie izolowane w sensie Hamminga. Optimum lokalne, które jest maksymalnie oddalone od optimum globalnego „ściąga" ku sobie rozwiązanie.



Rys. 9. Zwodnicza funkcja pułapkowa

Zostały zdefiniowane trzy warunki zwodniczości funkcji [22]:

1) $f(0) > f(l)$;
2) $f(l) > f(0) - [f(l-1) - f(1)]$;
3) $f(i) >= f(j)$, dla $[l/2] <= i <= l-1$ i $l - i <= j < i$.

Dwubiegunowe funkcje zwodnicze, dla testowania algorytmów poszukiwania minimum można konstruować poprzez składanie funkcji jednomodalnej, lub stosując współczynniki Walsha [22].

Kolejnym trudnym problemem jest zagadnienie długiej ścieżki [23, 21]. Są to unimodalne, pozornie łatwe funkcje, które prostemu algorytmowi typu wspinacz (*hillclimber*) gwarantują odnalezienie optimum, ponieważ każdy punkt przestrzeni leży na ścieżce prowadzącej do szukanego rozwiązania. Trudność polega na tym, że ścieżka jest długa i ze względu na wymagania czasowe *hillclimber*, postępujący stałymi krokami jest zupełnie bezużyteczny. Ścieżka P o kroku k to sekwencja punktów p_i:

$$\forall\, p_i, p_j \in P : d(p_i, p_j) \le k, \quad \text{dla} \quad |i-j| \tag{6}$$

gdzie $d(p_i, p_j)$ – odległość Hamminga między punktami p_i i p_j.

Dla funkcji testowych ścieżki konstruowane są rekurencyjnie, wychodząc od ścieżki o wymiarze l. Dla minimalnego kroku $k = 1$ możemy tego dokonać poprzez utworzenie dwóch kopii ścieżki, poprzedzając poszczególne elementy ciągiem '11' oraz '00'. Następnie łączy się pierwszą kopię z odwróconą drugą kopią za pomocą punktu rozpoczynającego się ciągiem '01'. Rozważmy następującą ścieżkę P_l o wymiarze $l = 2$, kroku $k = 2$ i długości $|P_l| = 4$: $P_l = ((10), (00), (01), (11))$. Konstruując ścieżkę o wymiarze l = 4 tworzy się kopię 00: ((0010), (0000), (0001), (0011)) oraz kopię 11: ((1110), (1100), (1101), (1111)), a łącząc je za pomocą punktu (0111) uzyskuje się następującą ścieżkę P_4: ((0010), (0000), (0001), (0011), (0111), (1111), (1101), (1100), (1110)) o długości $|P_4| = 2 |P_2| + 1 = 9$. Długość ścieżki rośnie ekspotencjalnie (z podstawą $\sqrt{2}$) wraz ze wzrostem rozmiaru zadania [23] zgodnie ze wzorem (6)

$$|P_l| = 3 \cdot 2^{\left|\frac{l-1}{2}\right|} - 1 \tag{7}$$

gdzie:

l – wymiar zadania,
$|P_l|$ – długość ścieżki.

Dla dużych wartości l, ścieżka zajmuje niewielką część przestrzeni poszukiwań (2^l) i zadanie sprowadza się do problemu „igły w stogu siana" (NIAH – *Needle-In-a-Haystack*). Trudność tego typu zagadnienia polega na odnalezieniu wyizolowanej ścieżki w dziedzinie funkcji, a następnie rozwiązaniu problemu „długiej ścieżki".

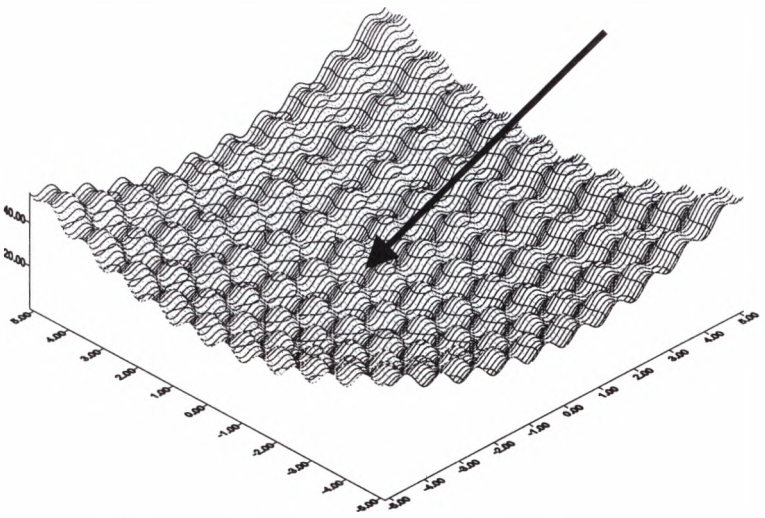
3.2. Funkcje testowe

Aby porównać różne algorytmy optymalizujące, używa się specjalnie dobieranych funkcji testowych. Pierwszy popularny zestaw testowy, dla zadań minimalizacji, złożony z pięciu funkcji, zaproponował De Jong [2]. Wraz z rozwojem metod obliczeniowych, zaczęto stosować coraz bardziej złożone funkcje testowe. Towarzyszą one międzynarodowym konkursom algorytmów optymalizujących (ICEO – *International Contest of Evolutionary Optimization)*. Pierwszy taki konkurs odbył się w 1996 roku w Japonii, w Nagoya. Algorytmy oceniane były w dwóch kategoriach: poszukiwania ekstremów funkcji rzeczywistych oraz problemów kombinatorycznych. Obu grupom zaproponowano zestaw zadań testowych [29]. Dla algorytmów z kategorii kombinatorycznych jest to zestaw problemów komiwojażera, natomiast programy z pierwszej grupy poszukują minimum pięciu różnych ciągłych funkcji testowych.

Do przedstawionych w tej pracy testów wybrano pięć funkcji ciągłych w pięcio- i dziesięciowymiarowych wersjach. Niektóre z nich pochodzą z łoża testowego pierwszego i drugiego konkursu ICEO, inne są określane jako szczególnie trudne do optymalizacji [19]. Wszystkie są multimodalne, a wartość minimum globalnego wynosi 0. Poniżej przedstawiono kilka charakterystycznych funkcji testowych w dwuwymiarowej wersji.

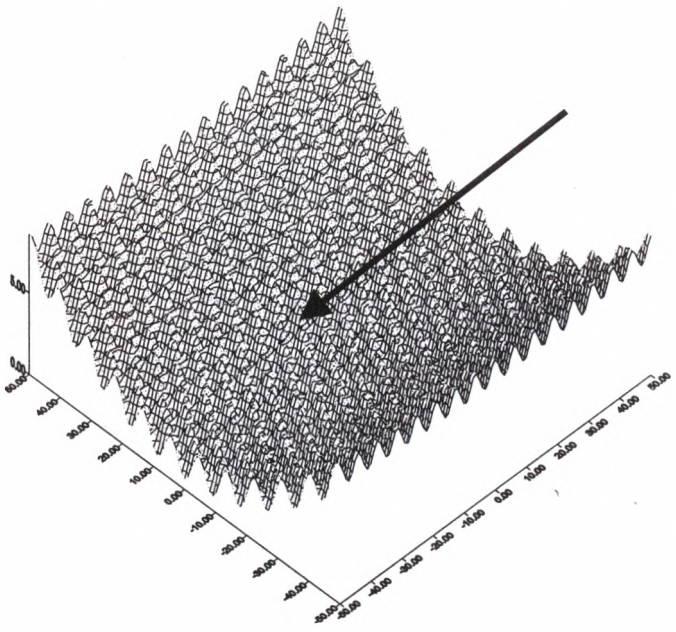
Funkcja Rastrigina (patrz rys. 10) dana jest wzorem

$$f(\overline{X}) = 3{,}0 \cdot n + \sum_{i=1}^{n} (x_i^2 - 3{,}0 \cdot \cos(2\Pi \cdot x_i)) \tag{8}$$



Rys. 10. Funkcja Rastrigina

Minimum globalne poszukiwane jest w przedziale: <– 5,12; 5,12>. Cechą charakterystyczną tej funkcji jest obecność wielu minimów lokalnych, których wartość zwiększa się wraz z oddalaniem od minimum globalnego, znajdującego się w punkcie (0,0,0...). Jest to funkcja z jednym łagodnym minimum lokalnym, która została „zaszumiona" równomiernie w całej dziedzinie. Amplituda szumów rośnie wraz z oddaleniem się od minimum globalnego. Funkcja Rastrigina po wygładzeniu stałaby się prostą, unimodalną, paraboliczną funkcją. Minimum globalne „dostrzegane" jest z każdego punktu dziedziny.



Rys. 11. Funkcja Griewangka

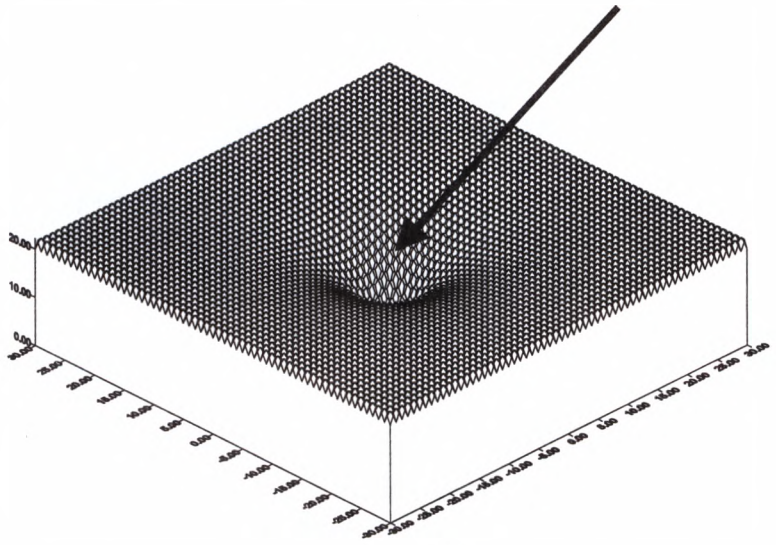
Minimum globalne funkcji Griewangka (rys. 11), danej wzorem

$$F(\bar{X}) = 1 + \sum_{i=1}^{n}\left(\frac{x_i^2}{400} - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)\right) \tag{9}$$

poszukiwane w przedziale $x_i \in$ <– 0150600; 600> leży w punkcie (0,0,...). Jest to również zaszumiona funkcja, o jednym wyraźnym minimum globalnym, które jest „wyczuwalne" z każdego punktu dziedziny. Minimum globalne leży w „długiej dolinie", a więc dodatkowym utrudnieniem jest tu problem „długiej ścieżki".

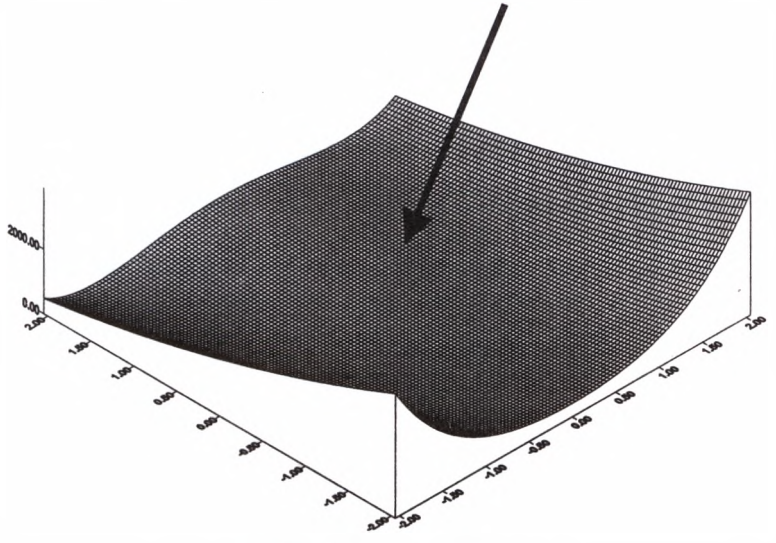
Funkcja Ackleya (rys. 12) zadana jest wzorem

$$f(\bar{X}) = 20 + e \cdot \exp\left(-0{,}2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\Pi \cdot x_i)\right) \tag{10}$$



Rys. 12. Funkcja Ackleya

Minimum poszukiwane jest w przedziale <– 30; 30>. Minimum globalne funkcji Ackleya, leży w punkcie (0,0,..). Poza nim dziedzina pokryta jest dużą ilością małych górek i dolinek. Mamy tu również do czynienia z problemem „długiej ścieżki", gdyż z większości punktów dziedziny do granic leja prowadzi długa, łagodna droga. W przeciwieństwie do funkcji Griewangka, w momencie zbliżania się do minimum globalnego funkcja staje się bardzo stroma.

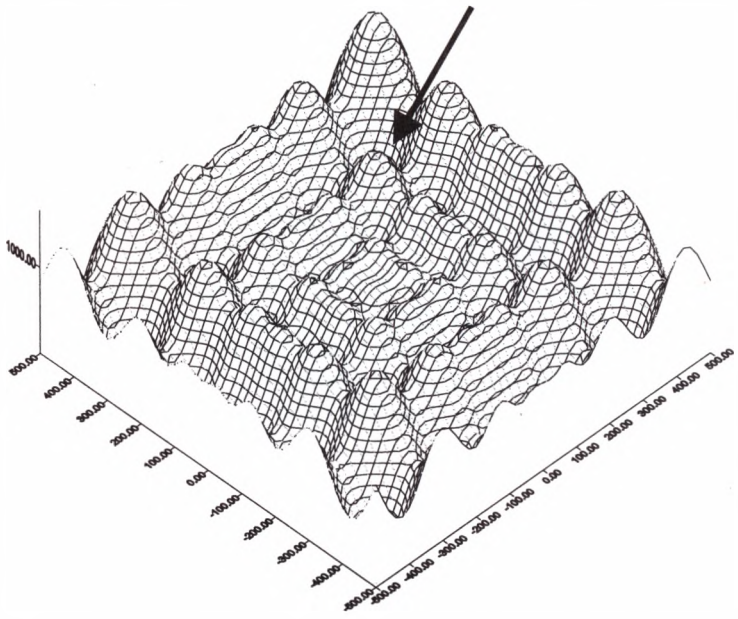


Rys. 13. Funkcja Rosenbrocka

Minimum globalne funkcji Rosenbrocka (rys. 13), opisanej równaniem

$$f(\bar{X}) = \sum_{i=1}^{n-1}(1 - x_i)^2 + 100 \cdot (x_{i+1} - x_i)^2 \tag{11}$$

poszukiwane w przedziale: <–2.048; 2.048> znajduje się w punkcie (1,1,...). Charakterystyczną cechą tej unimodalnej funkcji jest bardzo głęboka dolina, prowadząca do minimum. Jest to typowy problem „długiej ścieżki", z dodatkowo załamaną doliną.



Rys. 14. Funkcja Schwefela

Minimum globalne funkcji Schwefela (rys. 14), opisanej równaniem

$$f(\bar{X}) = 418{,}9829 \cdot n + \sum_{i=1}^{n} x_i \cdot \sin(\sqrt{|x_i|}) \tag{12}$$

poszukiwane w przedziale <–500; 500>, leży w punkcie (420.9686,420.9687,....). Jest to trudna, funkcja zwodnicza, o wielu minimach lokalnych. Daleko od minimum globalnego leży minimum lokalne, które „przyciąga" rozwiązanie. W przeciwieństwie do funkcji Rastrigina, Griewangka, Ackleya, Rosenbrocka, minimum globalne nie jest „wyczuwalne" z żadnego punktu dziedziny.

3.3. Wskaźniki używane do oceniania algorytmów

Dla potrzeb konkursu optymalizujących algorytmów ewolucyjnych (ICEO) zdefiniowano także kryteria oceny. Efektywność algorytmów porównywana jest na podstawie poniższych trzech indeksów [29].

1) ENES (*Expected Number of Evaluations per Succes*) – średnia ilość obliczeń funkcji konieczna do osiągnięcia sukcesu, czyli odnalezienia minimum funkcji z zadaną dokładnością. Obliczana jest jako iloraz NE, tj. całkowitej liczby wartościowań funkcji w 20 uruchomieniach programu, z takimi samymi parametrami i NS, czyli liczby sukcesów, tj. ilości uruchomień, w których uzyskano pożądaną wartość ekstremum VTR (*Value To Reach*).
2) BV (*Best Value*) – najlepsza wartość minimum, uzyskana w 20 uruchomieniach programu.
3) RT (*Relative Time*) – względny czas wykonywania obliczeń innych od wartościowania funkcji celu. Obliczany jest według wzoru: RT = (CT – ET)/ET, gdzie CT jest całkowitym czasem CPU, potrzebnym do wykonania 10 000 iteracji, a ET to czas CPU wykonania 10000 ewaluacji funkcji celu. RT jest nieużyteczny przy stosowaniu równoległych algorytmów.

4. Sposoby przeszukiwania przestrzeni rozwiązań i optymalne parametry AG i MD

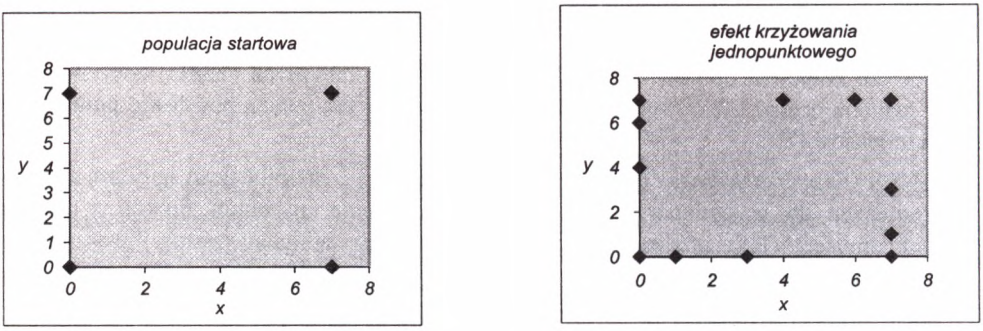
Do badań porównujących algorytm genetyczny (AG) i metodę cząstek (MD) wykorzystano standardowe wersje obu algorytmów. Niniejszy rozdział zawiera analizę obszaru przeszukiwań obu algorytmów oraz wyniki testów, służących doborowi optymalnych parametrów dla tych metod. Tak dobrane parametry zostały użyte do testów porównujących, których wyniki zawiera rozdział 5.

4.1. Heurystyki optymalizacyjne i dobór optymalnych parametrów dla AG

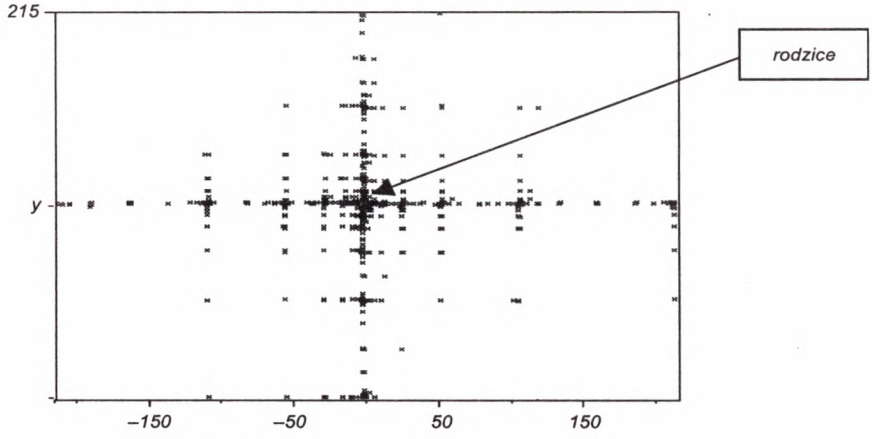
Podstawowy algorytm genetyczny podlega częstym modyfikacjom [35, 37, 40]. Stosuje się np. różne operatory selekcji i krzyżowania. Najprostszy operator krzyżowania, to krzyżowanie jednopunktowe. Dwa macierzyste ciągi kodowe dzieli się na podłańcuchy i wymienia się fragmenty kodów obu osobników. Punkt podziału generowany jest losowo.

Rysunek 15 ilustruje efekt krzyżowania jednopunktowego. Punkty potomne, definiujące obszar przeszukiwany przez algorytm, układają się wzdłuż boków prostokąta zdefiniowanego przez rodziców. Obszar wewnętrzny nie jest wówczas w ogóle przeglądany.

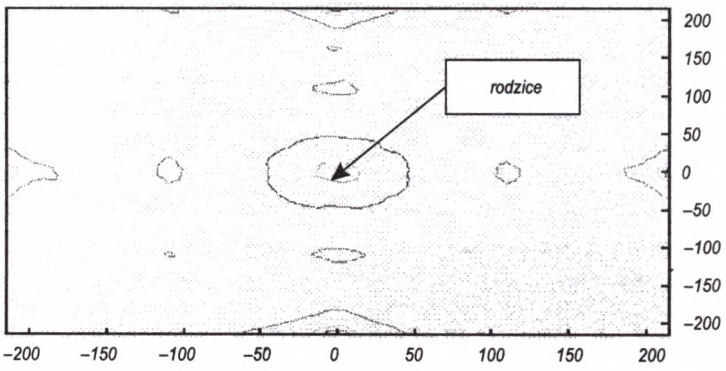
Rysunek 16 ilustruje rozkład punktów potomnych, uzyskanych za pomocą krzyżowania i średniej mutacji, startując od dwóch rodziców, umieszczonych w centrum obszaru. Wyraźnie widoczne są charakterystyczne krzyże oraz większe zagęszczenie potomków bliżej punktów rodzicielskich. Rysunek 17 przedstawia ilościowy rozkład punktów potomnych w całej dziedzinie.



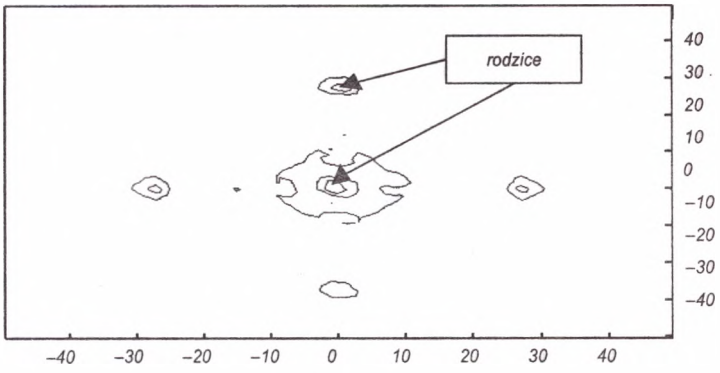
Rys. 15. Rozkład punktów potomnych dla krzyżowania jednopunktowego



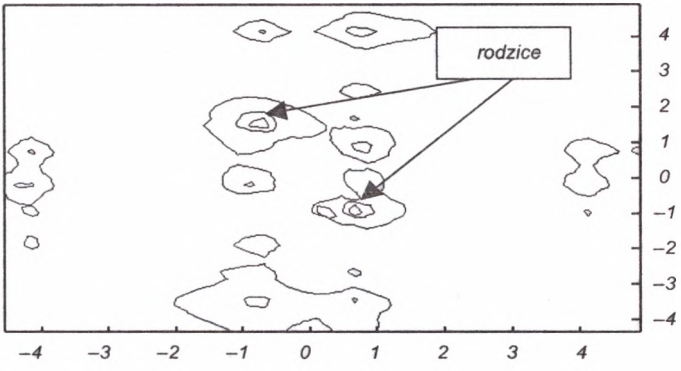
Rys. 16. Rozkład punktów potomnych dla krzyżowania z mutacją



Rys. 17. Obszar przeszukiwania całej dziedziny



Rys. 18. Obszary przeszukiwania części dziedziny (środkowy rejon)

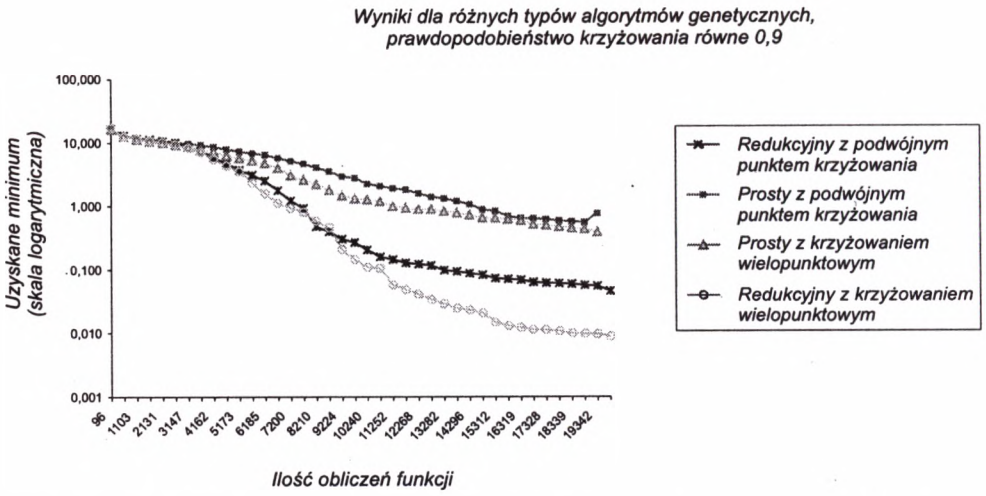


Rys. 19. Obszary przeszukiwania dziedziny w okolicy punktów rodzicielskich w powiększeniu

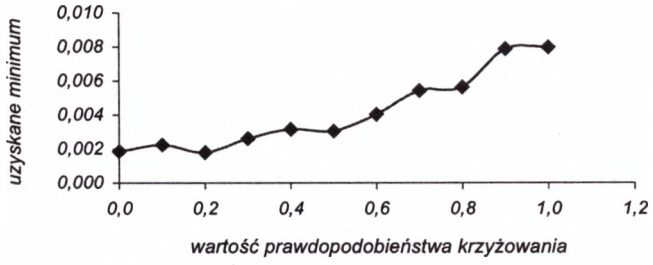
Rysunki 18 i 19 ilustrują rozkład punktów potomnych w kolejnych przybliżeniach. Układają się one w wyraźne klastry, ułożone wzdłuż linii prostopadłych, przechodzących przez punkty startowe, wokół których skupia się największa ilość potomków.

Jak wykazały testy [20], także wybór operatora krzyżowania istotnie wpływa na skuteczność algorytmu. Krzyżowanie wielopunktowe daje zwykle lepsze efekty. Okazało się także, że wprowadzenie redukcji takich samych osobników (algorytm redukcyjny), czyli zwiększenie różnorodności populacji, również polepsza efektywność. Przeprowadzono testy [20], przyjmując prawdopodobieństwo krzyżowania 0,9, natomiast prawdopodobieństwo mutacji 0,01. Testy przeprowadzono dla wybranych funkcji, przedstawionych w rozdziale 3. Najlepsze wyniki uzyskano dla algorytmu wykorzystującego wielopunktowe krzyżowanie i redukcję populacji. Jako przykład, na rysunku 20 przedstawione są rezultaty testów dla funkcji Ackleya.

Podobnie dla wszystkich wybranych funkcji testowych zostały dobrane optymalne wartości prawdopodobieństwa krzyżowania. Przykładowo dla funkcji Ackleya wynosi ona 0,2 (rys. 21). Z takim parametrem powtórzono test z rysunku 20 i uzyskano lepsze wyniki końcowe, co potwierdza słuszność takiej modyfikacji.



Rys. 20. Wpływ wyboru operatora krzyżowania oraz typu algorytmu na uzyskane minimum

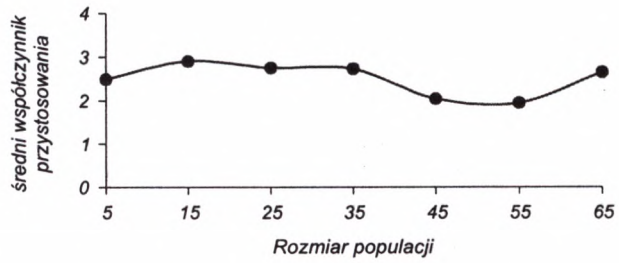


Rys. 21. Wpływ wartości prawdopodobieństwa krzyżowania na uzyskane minimum

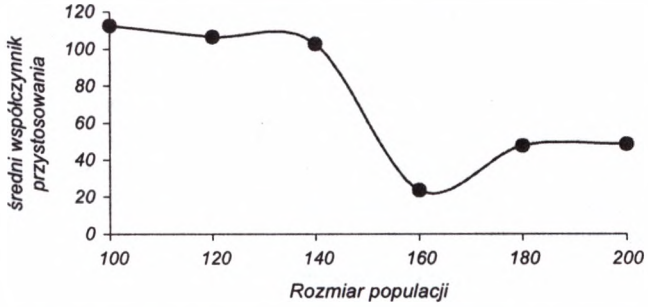
Liczebność populacji wpływa na efektywność algorytmu. Istnieje pewna optymalna liczba konieczna do znalezienia optimum. Mała populacja ogranicza różnorodność, a większa wydłuża z kolei czasy obliczeń. Testy przeprowadzone dla pięciu wybranych do badań funkcji wykazały [20], że dla każdej z nich istnieje pewna optymalna wielkość populacji. Na przykład dla funkcji Rastragina jest to liczba 50, natomiast dla Schwefela – 160.

Test, którego wyniki pokazuje rysunek 22 przeprowadzono dla pięciowymiarowej funkcji Rastragina. Przyjęto następujące parametry: ilość obliczeń funkcji = 100000, prawdopodobieństwo krzyżowania = 0,2, prawdopodobieństwo mutacji = 0,01. Wynik uśredniono dla 20 powtórzeń obliczeń.

Rysunek 23 przedstawia wyniki testu przeprowadzonego dla pięciowymiarowej funkcji Schwefela z zastosowaniem parametrów: ilość obliczeń funkcji = 100000, prawdopodobieństwo krzyżowania = 0,9, prawdopodobieństwo mutacji = 0,01. Wynik uśredniono dla 10 powtórzeń obliczeń.



Rys. 22. Wpływ wielkości populacji (ilości cząstek) na uzyskane minimum (funkcja Rastragina)



Rys. 23. Wpływ wielkości populacji (ilości cząstek) na uzyskane minimum (funkcja Schwefela)

Jak widać z rysunków 22 i 23 istnieje rozmiar populacji, dla którego algorytm osiąga najlepszy wynik. Widać stąd, iż z jednej strony istotną rzeczą jest zróżnicowanie populacji początkowej, z drugiej ograniczenie tego zróżnicowania do pewnej „perspektywicznej" grupy rozwiązań. Gwarantuje to najpełniejszą penetrację obszaru przeszukiwań i umożliwia znalezienie rozwiązania optymalnego.

4.2. Adaptacja metody MD do problemów znajdowania najlepszego rozwiązania

Drugą metodą użytą w testach porównujących jest metoda cząstek, opisana w podrozdziale 2.3. Istnieje wiele modyfikacji metody podstawowej. W niniejszym opracowaniu zastosowano algorytm uproszczony.

Konstrukcja sił oddziaływań między cząsteczkami

Obliczanie gradientu funkcji F występującego w równaniu

$$m_i \cdot \frac{\mathrm{d}V_i}{\mathrm{d}t} = -a \cdot \nabla F(X_i) - \lambda \cdot V_i + k \cdot f_{oddz.} \tag{13}$$

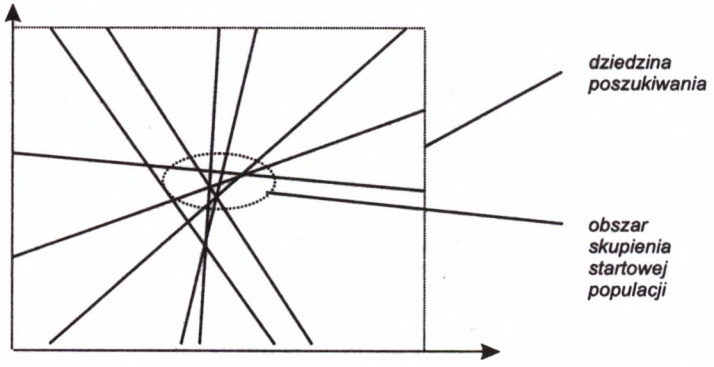
jest poważnym utrudnieniem w stosowaniu powyższej metody. Dlatego w przeprowadzonych testach [20] zastosowano zmodyfikowany algorytm MD, w którym całkowicie wyeli-

minowano siły gradientowe z równań ruchu. Zrezygnowano także z dwucząstkowego potencjału Lennarda-Jonesa. Siła oddziaływań między cząstkami f_{oddz} została zdefiniowana w następujący sposób [20]:

- jest siłą wyłącznie przyciągającą,
- jest niesymetryczna – dla każdej rozpatrywanej osobno pary cząstek *i, j*, cząstka x_i położona wyżej – tzn. $F(x_i) > F(x_j)$ jest przyciągana przez leżącą niżej (wartość siły może zwiększać się wraz ze wzrostem różnicy wysokości, zmniejszać lub pozostawać stała),
- dwie cząstki oddziałują ze sobą, jeśli odległość między nimi jest mniejsza od R_{max} (wartość siły może się zmniejszać wraz ze wzrostem odległości lub pozostawać stała).

Przeprowadzone testy wykazały [20], że najlepsze opcje programu to: niezależna od odległości i zwiększająca się wraz z różnicą wysokości siła i R_{max} równe średnicy rozpatrywanego obszaru zmienności funkcji. Zrezygnowanie z sił gradientowych i zastąpienie ich przez specjalną siłę oddziaływań bardzo upraszcza metodę, ale wymaga stosowania większej ilości cząstek. Optymalna sytuacja występuje wtedy, gdy każda cząstka ma w zasięgu swoich oddziaływań co najmniej jednego sąsiada w każdym kierunku ruchu, i dzięki sile oddziaływań rosnącej wraz z różnicą wysokości może „kierować się" w stronę największego spadku wartości funkcji $F(x)$.

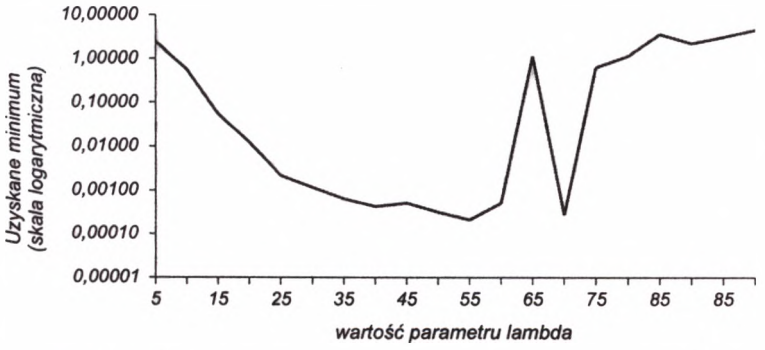
Pozytywny wpływ na znajdowanie minimum globalnego metodą cząstek ma ulepszenie polegające na niezmienianiu położenia cząstki, leżącej w danym kroku najniżej. Cząstka taka staje się silnym przyciągającym centrum dla wszystkich innych cząstek i przez to potrafi wyprowadzić je z minimów lokalnych. Wystarczy, że początkowe (lub w wyniku chaotycznych szybkich ruchów na początku symulacji) położenie którejś cząstki wypadło blisko minimum globalnego, a możemy być pewni, że reszta cząstek też je zauważy. Cząstki poruszają się wzdłuż linii prostych, wyznaczając obszary przeszukiwania dziedziny przez algorytm. Dlatego ważne jest ich początkowe rozmieszczenie. Rysunek 24 ilustruje sposób penetracji dziedziny funkcji przez algorytm, gdy populacja startowa skupiona jest w centrum dopuszczalnego obszaru.



Rys. 24. Przeszukiwanie obszaru dziedziny w MD, dla populacji startowej w postaci klastra położonego centralnie

Inne ulepszenia, takie jak: wprowadzenie współczynnika tarcia zależnego od wysokości, czy selektywna siła oddziaływań (tylko pomiędzy wylosowanymi parami cząstek) nie wpływają znacząco na wyniki poszukiwania [20].

Jednakowa dla wszystkich cząstek masa m, określa ich bezwładność i wraz z jej zmniejszaniem cząstki bardziej ulegają działającym siłom. Przedział czasowy Δt oznacza ile czasu przypada na pojedynczy krok symulacji. Im większy krok czasowy, tym większe odległości pokonuje pojedyncza cząstka w jednym kroku. Współczynnik oddziaływań k jest stałą proporcjonalności, z jaką siły oddziaływań występują w równaniu ruchu. Parametr ten jest bardzo ważny ze względu na rolę, jaką w programie pełni siła oddziaływań – to dzięki niej możliwa jest eksploracja dziedziny przez cząstki i dążenie do znajdowania coraz niżej położonych punktów. Współczynnik tarcia λ jest stałą proporcjonalności, z jaką siła tarcia występuje w równaniu ruchu. Od niego zależy szybkość wygasania ruchu cząstek. Dla zbyt dużych wartości λ, cząstki nie zdążą przeszukać dostatecznie szerokiej dziedziny rozwiązań, a przez to odnalezione minimum może być niewystarczającej jakości. Dla zbyt małych λ dobór optymalnych parametrów przeszukiwania może okazać się mało efektywny czasowo. Parametr R_{max} określa górną granicę odległości, przy której zachodzą oddziaływania. W zależności od problemu poszukiwanie może być bardziej (R_{max} małe) lub mniej (R_{max} duże) lokalne. Z testów dla rozpatrywanych problemów optymalizacji wynika, że parametr R być na tyle duży, aby wszystkie cząstki mogły naraz oddziaływać ze sobą. Należy więc zwiększać ten parametr wraz z wymiarem funkcji, gdyż dla większych wymiarów zwiększają się odległości (liczone jako pierwiastek z sumy kwadratów) i zmniejszać go wraz z czasem symulacji.

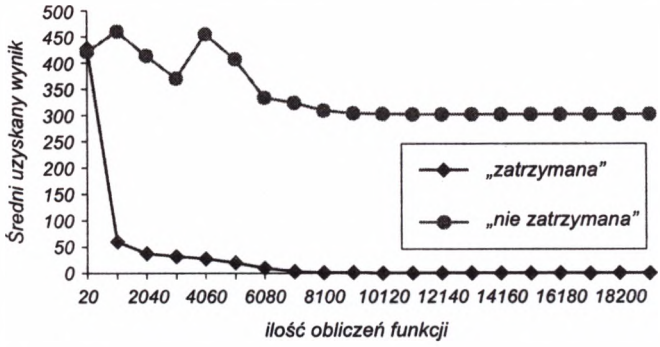


Rys. 25. Wpływ wartości parametru lambda na uzyskane rozwiązanie

Dla pięciu wybranych funkcji testowych zbadano wpływ wartości parametrów i wybranych opcji na uzyskany wynik. Poniższe wykresy ilustrują wyniki niektórych testów.

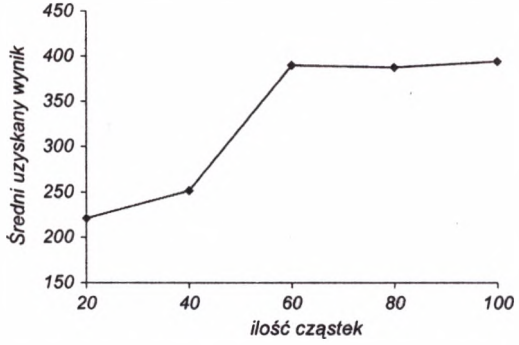
Test, którego wyniki przedstawia rysunek 25 przeprowadzono dla dwuwymiarowej funkcji Ackleya. Przyjęto następujące parametry rozmiar populacji (ilość cząstek) = 5, ilość obliczeń funkcji = 5000. Uzyskane minimum uśredniono dla 20 powtórzeń obliczeń. Najlepszy wynik uzyskano dla wartości współczynnika tarcia lambda równego 60.

Rysunek 26 przedstawia wyniki testu przeprowadzonego dla dwuwymiarowej funkcji Schwefela. Przyjęto następujące parametry: liczebność populacji = 10, ilość obliczeń funkcji = 20000, $k = 2000$, = 20. Wynik uśredniono dla 20 powtórzeń algorytmu. Bez zatrzymywania najniżej położonej cząstki, algorytm nie jest w stanie znaleźć minimum



Rys. 26. Wpływ opcji zatrzymywania najniżej położonej cząstki na uzyskany wynik

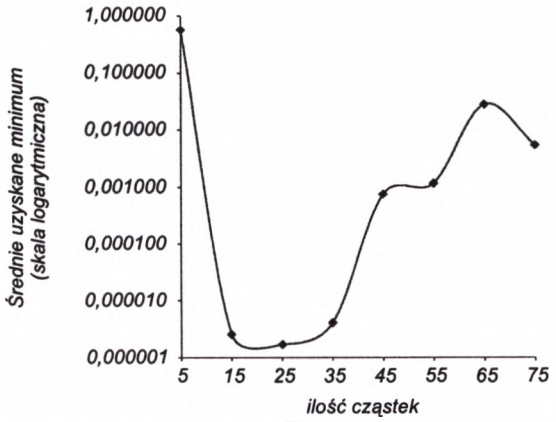
Rysunek 27 przedstawia wynik testu, przeprowadzonego dla pięciowymiarowej funkcji Schwefela. Przyjęto następujące wartości parametrów: ilość obliczeń funkcji = 100000, $k = 1000$, $\lambda = 6$. Wynik uśredniona dla 20 powtórzeń obliczeń. Najlepsze rezultaty uzyskano dla populacji równej 20.



Rys. 27. Wpływ wielkości populacji (ilości cząstek) na uzyskane minimum

Test, którego wynik przedstawia rysunek 28 przeprowadzono dla pięciowymiarowej funkcji Rastragina. Zastosowano wartości parametrów: ilość obliczeń funkcji = 100000, $k = 50$, $\lambda = 50$. Wynik uśredniono dla 20 powtórzeń obliczeń. Najlepszy rezultat uzyskano dla populacji o rozmiarze 20.

Do przeprowadzonych testów porównawczych, opisanych w kolejnym rozdziale, dla każdego z badanych problemów użyto najlepszych ustawień parametrów.



Rys. 28. Wpływ wielkości populacji (ilości cząstek) na uzyskane minimum

5. Porównanie algorytmu genetycznego i metody cząstek – wyniki testów

Algorytm genetyczny i metoda cząstek zostały poddane testom, mającym na celu porównanie ich efektywności w poszukiwaniu minimum funkcji rzeczywistych. Użyto funkcji testowych i wyznaczono indeksy opisane w rozdziale 3. Dla każdej funkcji testowej i każdego programu zostały dobrane najlepsze wartości parametrów. W algorytmie genetycznym zastosowano dwupunktowe krzyżowania oraz redukcję populacji. Dla metody cząstek wykorzystano modyfikacje opisane w rozdziale 4. Zgodnie z regułami konkursu ewolucyjnego testy przeprowadzono dla każdej z funkcji w pięcio- i dziesięciowymiarowej wersji. Do testów użyto programu implementującego oba algorytmy w wersji sekwencyjnej, działającego pod systemem operacyjnym WINDOWS'95 na komputerze typu PC z procesorem Pentium. Parametry programów użyte w poszukiwaniach minimum dla funkcji testowych zamieszczono w tabelach 1÷5.

Tabela 1

Parametry programów użyte w poszukiwaniach minimum dla funkcji Rastragina

Metoda cząstek (MD)	**Algorytm genetyczny (AG)**
Zestaw parametrów dla 5 wymiarów	*Zestaw parametrów dla 5 wymiarów*
k: 500 lambda: 100 populacja: 50	prawdop. krzyżowania: 0,9 prawdop. mutacji: 0,01 populacja: 160
Zestaw parametrów dla 10 wymiarów	*Zestaw parametrów dla 10 wymiarów*
k: 500 lambda: 100 populacja: 100	prawdop. krzyżowania: 0,9 prawdop. mutacji: 0,01 populacja: 200

Tabela 2

Parametry programów użyte w poszukiwaniach minimum dla funkcji Griewangka

Metoda cząstek (MD)	**Algorytm genetyczny (AG)**
Zestaw parametrów dla 5 wymiarów	Jednakowy zestaw parametrów Dla 5 i 10 wymiarów
k: 1000 lambda: 50 populacja: 50	prawdop. krzyżowania: 0,1 prawdop. mutacji: 0,01 populacja: 50
Zestaw parametrów dla 10 wymiarów	
k: 1000 lambda: 20 populacja: 50	

Tabela 3

Parametry programów użyte w poszukiwaniach minimum dla funkcji Ackleya

Metoda cząstek (MD)	**Algorytm genetyczny (AG)**
Dla funkcji Ackleya testy przeprowadzono z jednakowym zestawem parametrów dla 5 i 10 wymiarów	
k: 500 lambda: 10 populacja: 10	prawdop. krzyżowania: 0,1 prawdop.mutacji: 0,01 populacja: 10

Tabela 4

Parametry programów użyte w poszukiwaniach minimum dla funkcji Rosenbrocka

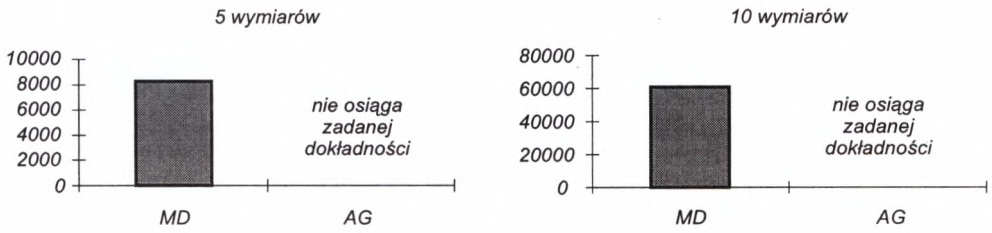
Metoda cząstek (MD)	**Algorytm genetyczny (AG)**
Zestaw parametrów dla 5 wymiarów	*Zestaw parametrów dla 5 wymiarów*
k: 50 lambda: 50 populacja: 30	prawdop. krzyżowania: 0,1 prawdop. mutacji: 0,01 populacja: 75
Zestaw parametrów dla 10 wymiarów	*Zestaw parametrów dla 10 wymiarów*
k: 50 lambda: 25 populacja: 30	prawdop. krzyżowania: 0,1 prawdop. mutacji: 0,01 populacja: 100

Tabela 5

Parametry programów użyte w poszukiwaniach minimum dla funkcji Schwefela

Metoda cząstek (MD)	**Algorytm genetyczny (AG)**
Zestaw parametrów dla 5 wymiarów	*Zestaw parametrów dla 5 wymiarów*
k: 3000 lambda: 6 populacja: 20	prawdop. krzyżowania: 0,9 prawdop. mutacji: 0,01 populacja: 160
Zestaw parametrów dla 10 wymiarów	*Zestaw parametrów dla 10 wymiarów*
k: 4000 lambda: 6 populacja: 40	prawdop. krzyżowania: 0,9 prawdop. mutacji: 0,01 populacja: 200

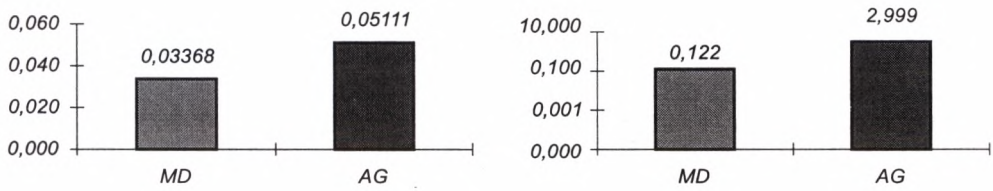
Jak widać z rysunku 29 algorytm genetyczny (AG) dla funkcji Rastrigina nie jest w stanie osiągać zadanej dokładności.



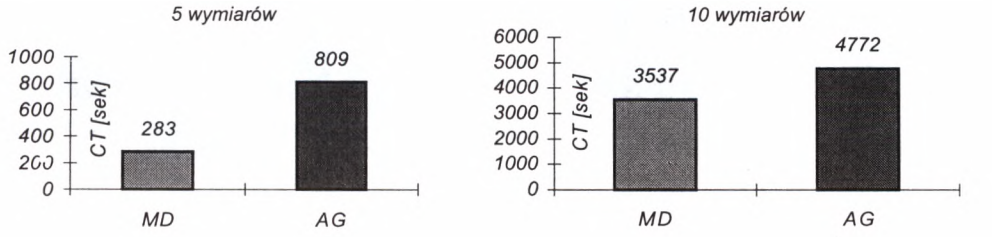
Rys. 29. Średnia ilość obliczeń funkcji potrzebna dla osiągnięcia sukcesu (funkcja Rastrigina)

Dla funkcji o wymiarze 10 wyniki dla obu algorytmów są o rząd wielkości gorsze niż dla funkcji pięciowymiarowej.

Dla problemu 5-wymiarowego algorytm genetyczny (AG) potrzebuje prawie 3 razy więcej czasu na znalezienie minimum, dla 10 wymiarów różnica ta wyraźnie się zmniejsza (patrz rys. 30÷32).



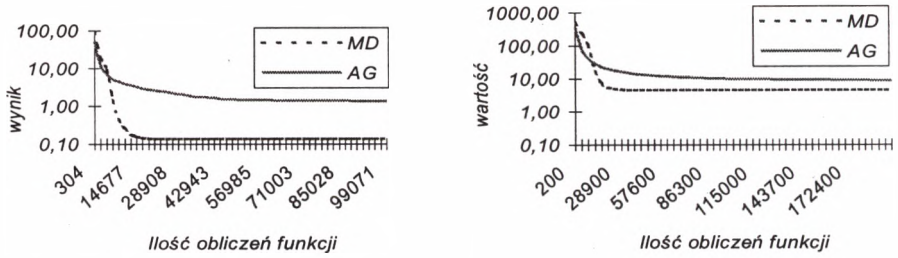
Rys. 30. Najlepsza uzyskana wartość (funkcja Rastrigina)



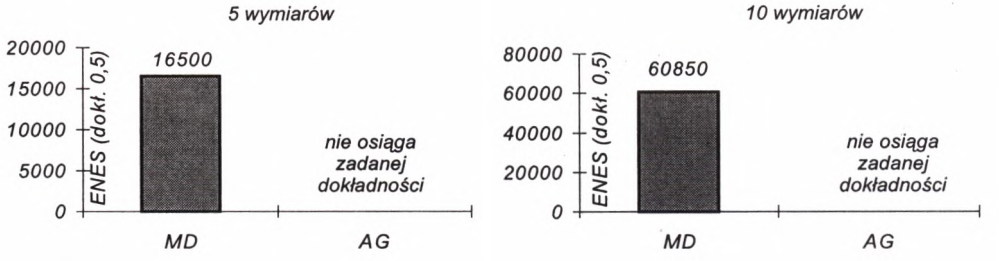
Rys. 31. Czas obliczeń dla 20 powtórzeń algorytmu w sekundach (funkcja Rastrigina)

Dla funkcji Rastrigina oba algorytmy szybko znajdują wartość bliską minimum globalnemu, po czym osiągnięty rezultat nie zostaje poprawiony (rys. 32).

Podobnie jak dla funkcji Rastragina, dla funkcji Griewangka w przeciwieństwie do algorytmu MD algorytm genetyczny (AG) nie był w stanie osiągnąć zadanej dokładności (rys. 33).

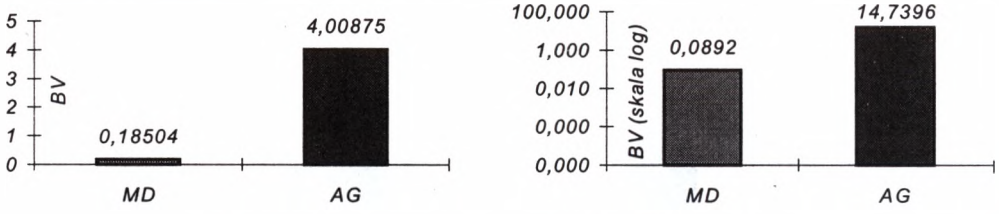


Rys. 32. Średni uzyskany wynik (skala logarytmiczna) w zależności od liczby obliczeń funkcji (funkcja Rastrigina)



Rys. 33. Średnia ilość obliczeń funkcji potrzebna dla osiągnięcia sukcesu (funkcja Griewangka)

Wartość BV jest najlepszą uzyskaną wartością, podczas 20 powtórzeń algorytmu, aż do uzyskania zadanej dokładności. Podobnie jak w poprzednim przypadku różnica na korzyść MD zmniejsza się dla funkcji 10-wymiarowej (rys. 34).

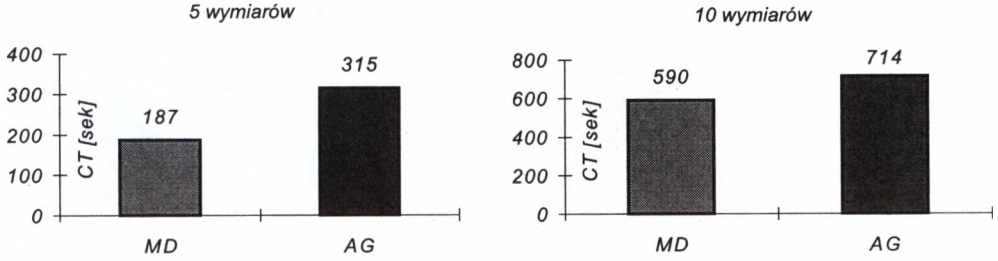


Rys. 34. Najlepsza uzyskana wartość (funkcja Griewangka)

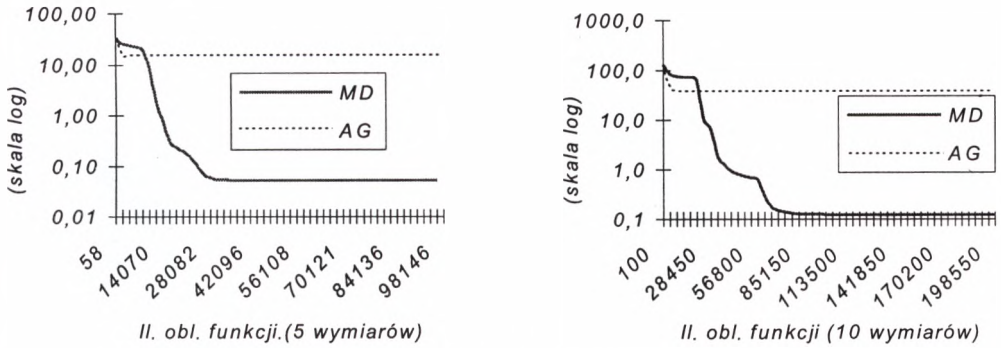
Jak widać z rysunku 35 z uwagi na skomplikowany wzór analityczny funkcji Griewangka, w porównaniu z innymi funkcjami czas obliczeń wzrósł.

Zwiększanie ilości obliczeń funkcji w przypadku algorytmu genetycznego (AG) nie wpływa na osiągnięcie lepszego wyniku (rys. 36). Wykresy dla metody cząstek ujawniają charakterystyczne dochodzenie do lepszego rozwiązania kolejnymi etapami.

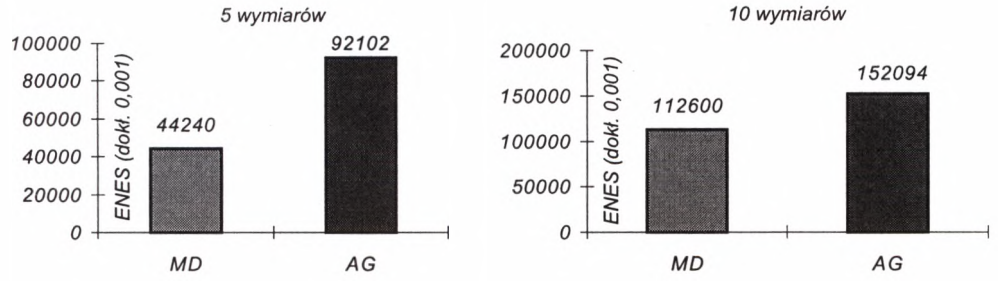
Funkcja Ackleya jest to jedyna funkcja (spośród użytych przez nas w teście), dla której algorytm genetyczny (AG) osiąga wymaganą dokładność (rys. 37). Metoda cząstek (MD) potrzebuje zdecydowanie mniej obliczeń funkcji.



Rys. 35. Czas obliczeń dla 20 powtórzeń algorytmu w sekundach (funkcja Griewangka 5 i 10 wymiarów)



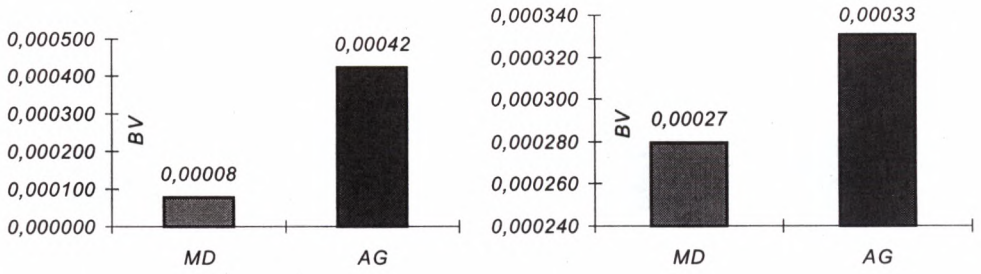
Rys. 36. Średni uzyskany wynik w zależności od liczby obliczeń funkcji (funkcja Griewangka)



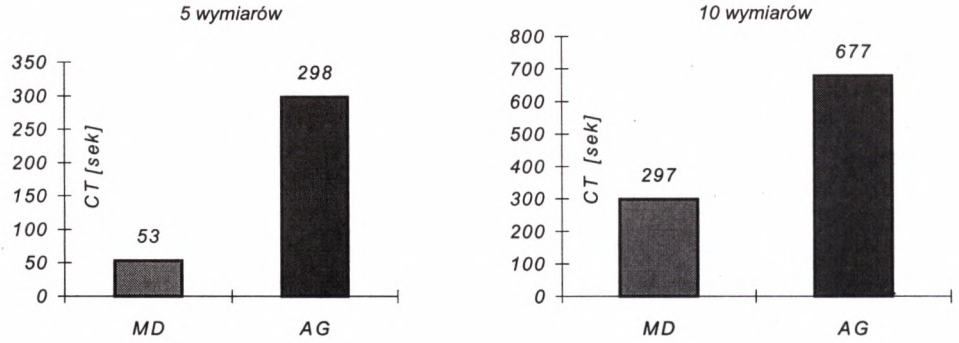
Rys. 37. Średnia ilość obliczeń funkcji potrzebna dla osiągnięcia sukcesu (funkcja Ackleya)

W obu przypadkach metoda cząstek (MD) daje rozwiązanie o rząd wielkości lepsze od rozwiązania algorytmu genetycznego (AG) – patrz rysunek 38.

Algorytm metody cząstek (MD) jest również zdecydowanie szybszy (czas dla 10 wymiarów równy jest czasowi algorytmu genetycznego (AG) dla 5 wymiarów) (rys. 39).

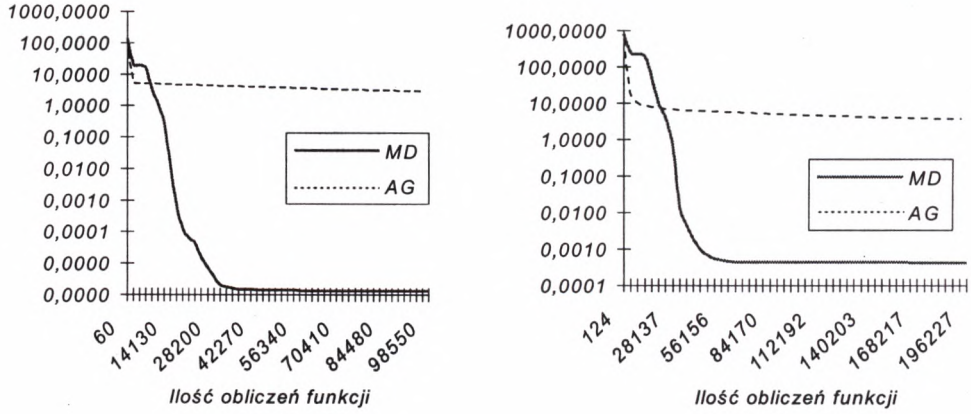


Rys. 38. Najlepsza uzyskana wartość (funkcja Ackleya)



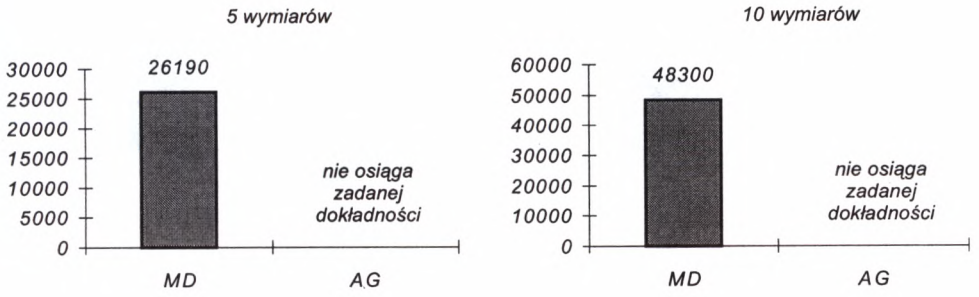
Rys. 39. Czas obliczeń dla 20 powtórzeń algorytmu w sekundach (funkcja Ackleya)

Algorytm genetyczny (AG) lokalizuje minimum po mniejszej ilości obliczeń funkcji niż metoda cząstek (MD), ale z mniejszą dokładnością (rys. 40).



Rys. 40. Średni uzyskany wynik (skala logarytmiczna) w zależności od liczby obliczeń funkcji (funkcja Ackleya)

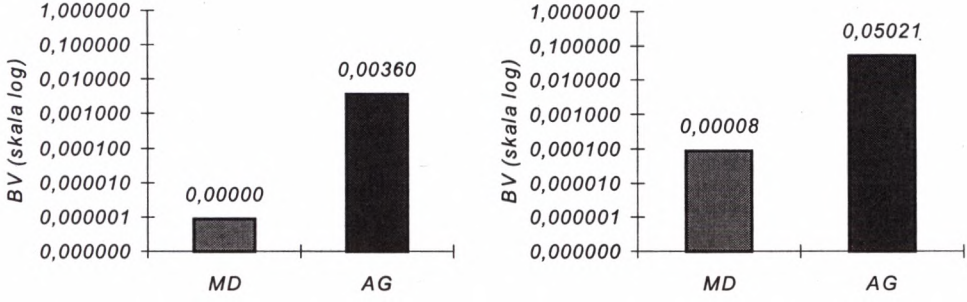
Dla kolejnej funkcji testowej – funkcji Rosenbrocka – algorytm genetyczny (AG), w przeciwieństwie do MD nie jest w stanie osiągać minimum z zadaną dokładnością (rys. 41).



Rys. 41. Średnia ilość obliczeń funkcji potrzebna dla osiągnięcia sukcesu (funkcja Rosenbrocka). Dla funkcji 5-wymiarowej zadana dokładność wynosi 0,00001, natomiast dla 10-wymiarowej 0,001

W obu przypadkach (5- i 10-wymiarowych) metoda cząstek (MD) daje lepsze o kilka rzędów wielkości rozwiązanie.

Czas obliczeń algorytmu genetycznego (AG) jest kilka razy dłuższy od czasu obliczeń metodą cząstek (rys. 42).



Rys. 42. Najlepsza uzyskana wartość (funkcja Rosenbrocka)

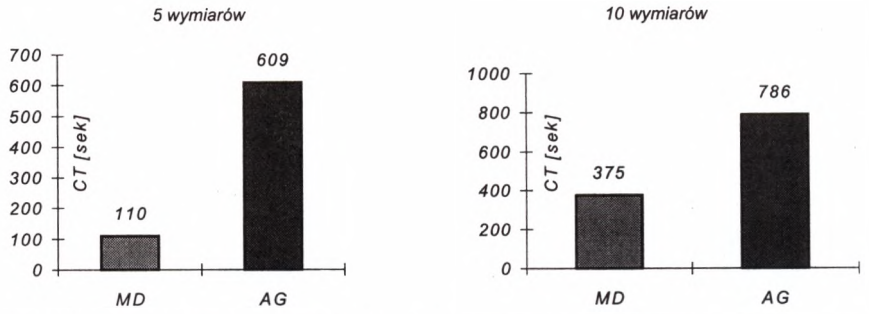
Algorytm genetyczny szybko osiąga wynik z małą dokładnością, metoda cząstek potrzebuje większej ilości obliczeń funkcji, ale daje lepszy wynik (rys. 43 i 44).

Dla funkcji Schwefela – jak widać na rysunku 45 – oba algorytmy nie mogły osiągnąć minimum z założoną dokładnością równą 1.

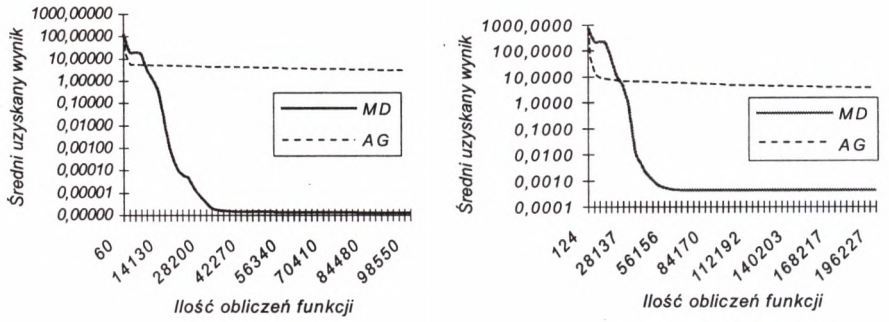
Jest to jedyna z funkcji (wśród użytych w teście), dla której algorytm genetyczny daje lepsze wyniki niż metoda cząstek (patrz rys. 46÷48).

Pojedynczy przebieg algorytmu genetycznego potrzebuje kilkakrotnie więcej czasu na obliczenia niż metoda cząstek.

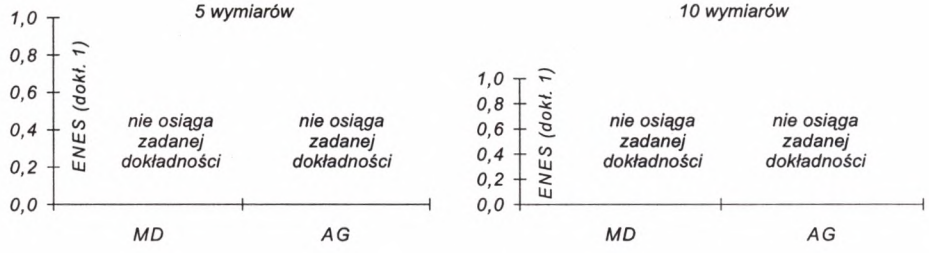
Utrzymująca się tendencja spadkowa wykresów dla obu algorytmów sugeruje, że zwiększenie ilości obliczeń funkcji może wpłynąć na poprawę osiągniętego wyniku.



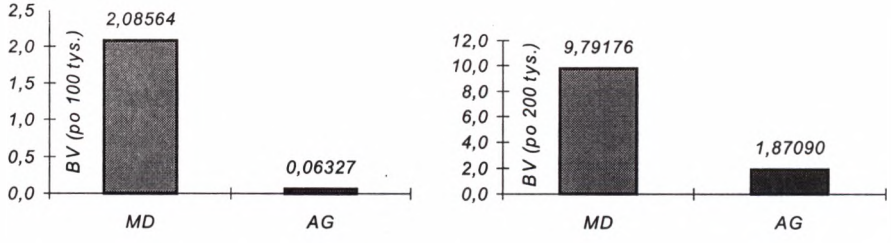
Rys. 43. Czas obliczeń dla 20 powtórzeń algorytmu w sekundach (funkcja Rosenbrocka)



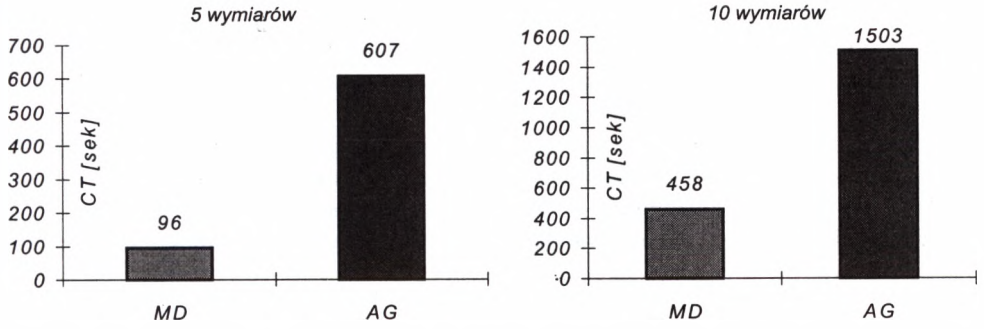
Rys. 44. Średni uzyskany wynik (skala logarytmiczna) w zależności od liczby obliczeń funkcji (funkcja Rosenbrocka)



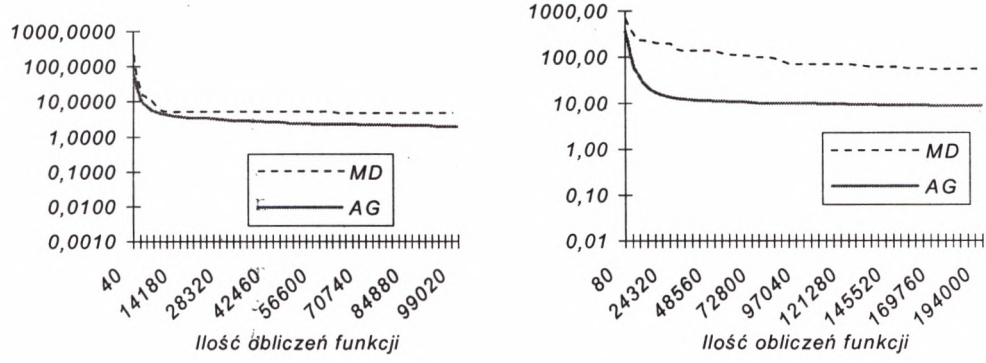
Rys. 45. Średnia ilość obliczeń funkcji potrzebna dla osiągnięcia sukcesu (funkcja Schwefela)



Rys. 46. Najlepsza uzyskana wartość (funkcja Schwefela)



Rys. 47. Czas obliczeń dla 20 powtórzeń algorytmu w sekundach (funkcja Schwefela)



Rys. 48. Średni uzyskany wynik (skala logarytmiczna) w zależności od liczby obliczeń funkcji (funkcja Schwefela)

6. Wnioski

Różnice w efektywności obu algorytmów wynikają głównie ze sposobu poszukiwania dziedziny. Algorytm genetyczny przeszukuje całą przestrzeń rozwiązań wzdłuż prostopadłych linii, ponieważ punkty potomne mają współrzędne odziedziczone po rodzicach, w wyniku operacji krzyżowania. W metodzie cząstek punkty poruszają się wzdłuż prostych łączących je, w kierunku lepszego rozwiązania, niejako „na azymut". Dlatego jest ona skuteczniejsza dla funkcji o wyraźnie wyróżnionym jednym minimum (Griewangka), a także dla zwodniczej funkcji, o kształcie doliny ze stromymi zboczami (Rosenbrocka). Ponadto algorytm genetyczny wymaga w takim przypadku znacznie większej liczby elementów populacji. Funkcja o charakterze zwodniczym (Schwefela), posiadająca kilka lokalnych minimów leżących daleko od minimum globalnego, wprowadza w błąd metodę cząstek – minima lokalne ściągają cząstki do siebie. Opcja zatrzymująca najniżej położoną cząstkę, umożliwiająca odnalezienie optimum dla innych funkcji, tu nie przynosi żadnych efektów. Algorytm genetyczny, dzięki skokowym zmianom położenia punktów, jest mniej podatny na takie pułapki. Szybciej zbiega się w okolice minimum. Dla danych rodziców punkty potomne

są zdeterminowane, niezależne nawet od rozwiązywanego problemu, jedynie wybór rodziców uwarunkowany jest wartością funkcji przystosowania. Natomiast w metodzie cząstek przemieszczanie cząstek jest ściśle uzależnione od wartości minimalizowanej funkcji i odbywa się zawsze w kierunku lepszego rozwiązania.

Metoda cząstek jest dobrym narzędziem do poszukiwania optimum multimodalnych, wielowymiarowych funkcji. Przykładem może być zastosowanie dynamiki molekularnej do ekstrakcji cech, czyli transformacji wielowymiarowej przestrzeni do przestrzeni 2- lub 3-wymiarowej, w celu wizualizacji danych [5]. Dokonuje się tego przez minimalizację kryterium zależnego od odległości między cząstkami w obu przestrzeniach. Metoda cząstek efektywniej poszukuje minimum funkcji ciągłej ze względu na analogię do analitycznych metod gradientowych. Wyjątek stanowi wspomniana funkcja Schwefela, ze względu na jej zwodnicze minima. Poszukiwanie uwarunkowane gradientowo (w kierunku „lepszego" sąsiada) prowadzi do utknięcia algorytmu w minimum lokalnym. Lepsze efekty daje tu zastosowanie algorytmu genetycznego, który bardziej nadaje się do rozwiązywania problemów natury kombinatorycznej. Być może dobrym rozwiązaniem byłoby zastosowanie algorytmu hybrydowego: AG zastosowany w pierwszym etapie pozwoliłby odnaleźć okolice globalnego minimum, a dalsze poszukiwania mogłyby być prowadzone przy pomocy metody cząstek.

Jak widać, w przypadku problemu poszukiwania najlepszego rozwiązania ważny jest odpowiedni dobór metody, w zależności od klasy i rodzaju problemu. Dobre efekty mogą przynieść algorytmy hybrydowe, w których metody naturalne stosowane są w pierwszej fazie dla odnalezienia przybliżonych rozwiązań. W przypadku funkcji wielomodalnej zastosowanie w pierwszym etapie algorytmu genetycznego pozwoli zlokalizować okolice minimów funkcji. Po takim przygotowaniu można zastosować inną metodę – np. analityczną. Dla funkcji wielomodalnych, „zaszumionych" lepszym rozwiązaniem będzie zastosowanie w pierwszej fazie dynamiki molekularnej. W przypadku, gdy charakterystyka funkcji, której minimum poszukujemy, jest nieznana, można stosować różne heurystyki: początkowo AG, a następnie MD lub w odwrotnej kolejności.

Dla obu omawianych metod istotne znaczenie ma dobór populacji początkowej. Algorytm genetyczny w pierwszym etapie sprawdza wiele różnych rozwiązań, a następnie dokładniej przegląda potencjalnie „lepsze" rejony. Jeśli początkowa populacja jest zróżnicowana i zawiera całościowy przekrój dziedziny, wówczas algorytm ma większe szanse na odnalezienie najlepszego rozwiązania. Źle przygotowana populacja startowa może sprawić, że pewne obszary dziedziny nie będą badane. Przykładem są testy [21], przeprowadzone dla dwubiegunowej funkcji zwodniczej, z wykorzystaniem zmodyfikowanego algorytmu genetycznego, w którym elementy populacji osadzone zostały na siatce, a operacja krzyżowania dozwolona jest jedynie między osobnikami sąsiadującymi [21, 31]. Pokazują one, że o skuteczności działania algorytmu decyduje rozkład populacji początkowej. Dla populacji wygenerowanej w sposób losowy algorytm odnajduje minimum zwodnicze zamiast globalnego.

Dla dynamiki molekularnej dobór populacji początkowej jest również zagadnieniem istotnym. Jeśli np. cząstki skupione są w centralnym obszarze dziedziny, skrajne rejony będą w niewielkim stopniu przeszukiwane. Jeśli tam usytuowane jest szukane minimum, pozostanie ono tym trudniejsze do odnalezienia, im dalej od startowego klastra jest położone (patrz funkcja Schwefela). Wynika to z faktu, że cząstki poruszają się wzdłuż linii prostych, w kierunku minimum, tworząc wiązkę rozproszoną, co ilustruje rysunek 24 z podroz-

działu 4.2. Wzrost temperatury układu wzmagający ruch cząstek może spowodować rozszerzenie obszaru przeszukiwań.

Jeśli charakter badanej funkcji jest znany, można łatwo dobrać najodpowiedniejszą metodę optymalizacyjną. Najczęściej spotykamy się jednak z przeciwną sytuacją, kiedy nie mamy żadnych informacji o przebiegu funkcji oraz o położeniu minimum. Metody naturalne są wówczas dobrym wyjściowym narzędziem do szukania najlepszego rozwiązania. Na nich mogą się opierać heurystyki służące do wstępnej analizy. Przy ich pomocy można gromadzić pewną wiedzę, aby następnie wykorzystać ją w rozwiązaniu problemu, czy to modyfikując algorytm, dopasowując go do problemu, czy konstruując nowy. Algorytmy zastosowane w pracy, to algorytmy standardowe i można je poddawać pewnym modyfikacjom, w zależności od charakteru problemu.

Heurystyki naturalne, do których zaliczają się oba omawiane w pracy algorytmy, uzupełniają się wzajemnie z metodami klasycznymi i dlatego razem stanowią kompletne narzędzie do poszukiwania najlepszego rozwiązania. Metody klasyczne zawodzą przy optymalizacji funkcji wielowymiarowych, natomiast lepiej nadają się do szukania ekstremum funkcji jednomodalnych. W rozwiązywaniu zadań kombinatorycznych częściej stosuje się metody heurystyczne. Pewien wyjątek stanowi problem komiwojażera, dla którego zostało wypracowane klasyczne podejście, przynoszące lepsze rezultaty niż heurystyki naturalne [46]. Nie jest to jednak możliwe, gdy problem jest mało zbadany. Wtedy bardziej przydatne są metody oparte o heurystyki naturalne. Służą one do wypracowania algorytmu specjalistycznego poprzez analizę odpowiedzi algorytmu heurystycznego na różnego rodzaju wektory wejściowe, których elementami są parametry heurystyki.

Literatura

[1] Krawczyk S.: *Programowanie matematyczne*. Warszawa, PWE 1980

[2] Goldberg D.E.: *Algorytmy genetyczne i ich zastosowania*. tłum. z j. ang. Warszawa, WNT 1995

[3] Battiti R., Tecchiolli G.: *Simulated Annealing and Tabu Search in the Long Run: a Comparison on QAP Tasks*. Computers and Mathematics with Applications, 1994, 28(6)

[4] Ingber L.: *Simulated annealing: Practice versus theory*. J. Math. Comput. Modelling, 1993, 18(11), 29

[5] Dzwinel W.: *Informatyczne problemy i perspektywy symulacji metodą cząstek*. Zeszyty Naukowe AGH, Rozprawy i monografie, 50, 1996

[6] Dzwinel W.: *Virtual particles and search for global minimum*. Future Generation Computer Systems, 12, 1997

[7] Tadeusiewicz R.: *Sieci neuronowe*. Warszawa, Akademicka Oficyna Wydawnicza RM 1993

[8] Hertz J., Krogh A., Palmer R.G.: *Wstęp do teorii obliczeń neuronowych*. tłum. z j. ang., Warszawa, WNT 1993

[9] Dorigo M., Maniezzo V., Colorni A.: *The Ant System: Optimization by a colony of cooperating agents*. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26, No. 1, 1996

[10] Battiti R., Tecchiolli G.: *Local Search with Memory: Benchmarking RTS.* Operations Reserch Spectrum 1995

[11] Sloot P.M.A., Kaandorp J.A, Schoneveld A.: *Dynamic Complex System (DCS): a new approach to parallel computing in computional physics.* Technical Report, University of Amsterdam, grudzień 1995

[12] Mahfoud S.W., Goldberg D.E.: *Parallel recombinative simulated annealing: A genetic algorithm.* Parallel Computing, 21, 1995

[13] Kirkpatrick S., Gellat C.D., Vecchi M.P.: *Optimization by Simulated Annealing.* Science, 220, 1984

[14] Dueck G., Scheuer T.: *Threshold Accepting:A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing.* Jurnal of Comput. Phys., 90, 1, 1990

[15] Ingber L.: *Adaptive Simulated Annealing (ASA): Lessons Lerned.* Control and Cybernetics, 1995

[16] Andricioaei I.A., Straub J.E.: *Finding the Needle in the Haystack: Algorithms for Conformational Optimization.* Computers in Physics, 10, Sep/Oct 1996

[17] Spears W.M.: *An overview of evolutionary computation.* European Conference of Machine Learning, 1993

[18] Harik G., Cantu-Paz E., Goldberg D., Miller B.L.: *The Gambler's Ruin Problem, Genetic Algoritmand the Sizing of Population*, IlliGAL web page

[19] Potter M.A.,De Jong K.A.: *A cooperative Coevolutionary Approach to Function Optimization.* The Third Parallel Problem Solving From Nature, Jerusalem, 1994

[20] Rozmus K., Sołtysiak J.: *W poszukiwaniu minimum globalnego.* Praca magisterska (AGH, Wydział Elektrotechniki, Automatyki Informatyki i Elektroniki), Kraków, czerwiec 1998

[21] Łaguza M.: *Problemy GA-trudne.* Praca magisterska (AGH, Wydział Elektrotechniki, Automatyki Informatyki i Elektroniki), Kraków, wrzesień 1998

[22] Goldberg D.E., Deb K., Horn J.: *Massive Multimodality, Deception and Genetic Algoritms.* IlliGAL Report No. 92005, kwiecień 1992

[23] Horn J., Goldberg D.E., Deb K.: *Long Path Problem.* The Third Conference on Parallel Problem Solving from Nature, Jerusalem, październik 1994

[24] Price K., Storm R.: *Ewolucja różnicowa.* Software, czerwiec 1997

[25] Strona WWW http://solon.cma.univie.ac.at/~neum

[26] Strona WWW http://gal4.ge.uiuc.edu/cgi-bin/orderform/

[27] Strona WWW http://cs.und.ac.za/~ikram/Research/

[28] Strona WWW http://www.wins.unva.nl/research/qwrs/introalg.html

[29] Strona WWW http://iridia.ulb.ac.be/Langerman/ICEO.html

[30] Strona WWW http://www.es.umass.edu/mie/

[31] Strona WWW http://www.abafi/~atorn/Globopt.html

[32] Broda A.: *Relacje przestrzenne i lokalność interakcji pomiędzy osobnikami populacji w algorytmach genetycznych.* Praca magisterska (AGH, Wydział Elektrotechniki, Automatyki Informatyki i Elektroniki), Kraków, wrzesień 1996

[33] Zieliński K. i inni: *Środowiska programowania rozproszonego w sieciach komputerowych.* Kraków 1994

[34] Campanini R., Di Caro G., Villani M., D'Antone I., Giusti G.: *Parallel architectures and intrinsically parallel algorithms: genetic algorithms*. International Journal of Modern Physics C, 5, 11, 1994
[35] http://www.aic.nrl.navy.mil/galist/
[36] http://gal4.ge.uiuc.edu/goldberg/d-goldberg.html/
[37] http://www-iligal.ge.uiuc.edu/%7Ebmiller
[38] Goldberg D.E.: *The design of innovation: lessons from genetic algoritms, lessons for the real world*. IlliGAL Report No. 98004, luty 1998
[39] Cantu-Paz E.: *A summary of research on parallel genetic algorithms*. IlliGAL Report No. 95007, 1995
[40] http://www.aic.nrl.navy.mil/galist/src/
[41] http://alumni.caltech.edu/~ingber/
[42] http://www.ingber.com/ASA_README.html
[43] http://www.wins.uva.nl/research/pwrs/projects/DynCompSys.html
[44] Fox G.C.: *Parallel Computers and Complex Systems*. Complexity International, 1, kwiecień 1994; dostępne przez: http://www.csu.edu.au/ci/vol1/Geoffrey.Fox/
[45] Findeisen W., Szymanowski J., Wierzbicki A.: *Metody obliczeniowe optymalizacji*. Warszawa, Wydawnictwa Politechniki Warszawskiej 1973
[46] Gajęcki M.: *Algorytm rozwiązujący problem komiwojażera i jego implementacja na różnych architekturach*. Praca doktorska (AGH, Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki), Kraków, 1996

Recenzent
prof. dr hab. inż. Edward Nawarecki