AUTOMATYKA • 2007 • Tom 11 • Zeszyt 1–2

Piotr Kadłuczka*, Jacek Piwowarczyk*, Wojciech Chmiel*

Współbieżny algorytm ewolucyjny wykorzystujący mechanizm samoadaptacji

1. Wprowadzenie

Inspiracją podjęcia rozważań na temat inteligentnych systemów agentowych były prace nad równoległymi wielopopulacyjnymi algorytmami ewolucyjnymi. W ramach prowadzonych badań nad efektywnością metod ewolucyjnych [1], analizując sposób tworzenia tych algorytmów, doszliśmy do wniosku, że zestawienie elementów konstrukcyjnych algorytmu jest dokonywane w dużej mierze intuicyjnie, na podstawie doświadczenia twórców (własnego oraz zaczerpniętego z literatury), a następnie weryfikowane eksperymentalnie.

W powyższym sposobie działania, problemem jest mnogość możliwych koncepcji każdego elementu algorytmu. Należy określić: postać rozwiązania (reprezentacja, sposób kodowania) i funkcji oceny przystosowania, sposób przetwarzania populacji (generacja populacji startowej, pokolenie, selekcja, elita), typy i mechanizm użycia operatorów genetycznych (krzyżowania, mutacji, dodatkowych np. optymalizacji lokalnej), kryteria zakończenia obliczeń, restartu, odświeżania populacji oraz wiele innych. Mechanizmy te opisane są wieloma parametrami (np. wielkość populacji, elity, prawdopodobieństwa użycia operatorów), których prawidłowy dobór wartości powinien być przeprowadzony dla reprezentatywnego zbioru zadań testowych (indywidualnie dla każdego zadania lub wszystkich razem). Wymagane jest też wielokrotne powtórzenie każdego eksperymentu, gdyż jedynie statystyczna ocena efektywności jest miarodajna dla metod losowych, do których należą także EA (*Evolutionary Algorithm*).

Przedstawiona metodyka prowadzenia badań skutkuje dużym nakładem pracy nawet przy wprowadzeniu niewielkich zmian w algorytmie, ze względu na występującą „interferencję" wszystkich jego elementów. Zamiar przerzucenia odpowiedzialności za dobór swoich elementów konstrukcyjnych i wartości parametrów na sam algorytm nie jest podejściem nowym. Zostało ono przedstawione w 1985 r. przez M.L. Cramera pod nazwą programowania genetycznego GP (*Genetic Programming*), a następnie ponownie odkryte i spopularyzowane przez J. Koza w 1992. GP to metoda automatycznego tworzenia programów komputerowych w oparciu o wysokopoziomową definicję zagadnienia, które ma zostać rozwiązane. Mimo że wielokrotnie powracano do tej koncepcji, słabość teorii oraz ograniczenia

* Katedra Automatyki, Akademia Górniczo-Hutnicza w Krakowie

wynikające z mocy obliczeniowej maszyn cyfrowych nie pozwoliły w pełni jej zrealizować. Pewnym odpryskiem metody jest ewolucyjny sposób doboru parametrów EA, przy wykorzystaniu chromosomu (rozwiązania) zawierającego zakodowane parametry.

Podstawową trudnością realizacji szeroko zakrojonej koncepcji GP jest znaczne rozszerzenie eksplorowanej przestrzeni. Jeśli rozważanym problemem jest zagadnienie NP-trudne (gdyż to one wymuszają użycie metod przybliżonych), to liczność przeszukiwanej przestrzeni rozwiązań jest dodatkowo multiplikowana przez liczbę możliwych do rozważenia elementów konstrukcyjnych (wymiennie lub łącznie) w każdym miejscu algorytmu oraz liczbę wartości przyjmowanych przez każdy parametr (zależną od zakresu i dyskretyzacji). Takie rozszerzenie przeszukiwanej przestrzeni uniemożliwia dokonania systematycznego przeglądu, w celu znalezienia optymalnych elementów i wartości parametrów algorytmu, co wymusza stworzenie efektywnej metody (zachłannej) prowadzącej do rozwiązań suboptymalnych. W przypadku wielu strategii przeszukiwania, także ich wybór chcieliśmy powierzyć algorytmowi ewolucyjnemu, co doprowadziło nas do koncepcji inteligentnych systemów agentowych (IMAS). Oczywiście można ją opisać w oparciu o pojęcia w nomenklaturze równoległych, wielopopulacyjnych EA, ale pewne elementy wykraczające poza ten obszar doskonale pasują do IMAS.

2. Wielowątkowy algorytm ewolucyjny a system wieloagentowy

Podstawowym zarzutem podnoszonym przez krytyków technologii agentowej jest argument, że systemy te mogą być opracowane i zaimplementowane także w inny sposób. Podejście to jest według nich modą, w której, wykorzystując zbiór własnych pojęć, jedynie w inny sposób opisuje się działanie systemu. Trudno się z tą krytyką nie zgodzić, ale z drugiej strony ten sposób opisu jest bardziej czytelny, szczególnie w przypadku bardzo złożonych systemów. Także w przypadku opisu koncepcji naszego algorytmu można ją opisać jako hybrydowy, współbieżny, samoadaptacyjny, wielopopulacyjny algorytm ewolucyjny.

2.1. Agent

Agent, jako pojęcie podstawowe dla systemów wieloagentowych jest u nas rozumiany jako jednostka obliczeniowa (wątek, program komputerowy realizujący algorytm ewolucyjny) działająca w określonym środowisku (przestrzeń rozwiązań, elementów konstrukcyjnych i parametrów algorytmu), cechująca się autonomią, elastycznością i stosująca pewną wiedzę dla realizacji postawionych celów (optymalizacja elementów i parametrów algorytmu oraz poszukiwanie najlepszego rozwiązania zagadnienia).

Lista cech przypisywanych agentom jest długa, co prowadzi do zróżnicowanych definicji agenta i systemów agentowych. Mimo niedoskonałości podziału i klasyfikacji, zbiór cech przytoczymy za artykułem M. Paprzyckiego [2]:

- reaktywność (*reactiveness*),
- ukierunkowanie na osiąganie celów (*goal orientation*),
- autonomia (*autonomy*),
- umiejętność dostosowania (*adaptivity*),
- umiejętność uczenia (*learning ability*),

- umiejętność porozumiewania się (*ability to communicate*),
- umiejętność współdziałania (*capacity for cooperation*),
- rozumowanie oparte o zgromadzoną wiedzę (*reasoning based on collected knowledge*),
- umiejętność przemieszczania się (*mobility*),
- bycie godnym zaufania (*reliability*),
- oddziaływanie (*interactivity*),
- umiejętność przewidywania (*proactivity*),
- umiejętność rozumowania (*capacity for reasoning*),
- inteligencja (*intelligence*).

Przyjmując bardziej precyzyjną definicję [3], agentem Λ nazywamy trójkę

$$\Lambda = (A, S, F \subset S \times A \times S), \tag{1}$$

gdzie:

A – zbiór akcji/decyzji (skończony),
S – skończony zbiór stanów agenta,
F – funkcja przejścia pomiędzy stanami.

Relacja F określa przejście pomiędzy stanami w wyniku wykonania akcji. Zachowanie związków przyczynowo-skutkowych (jednoznaczność) wymaga, aby prawdziwa była następująca implikacja

$$(s, a, s_1) \in F \wedge (s, a, s_2) \in F \Rightarrow s_1 = s_2.$$

Relacja F w sposób jednoznaczny, wiąże stany z akcjami, określając, do jakiego stanu prowadzi podjęcie danej decyzji w danym stanie oraz wyznacza zbiór stanów osiągalnych ze stanu aktualnego.

2.2. System wieloagentowy

Systemem agentowym Ω nazywany nazywany następującą dwójkę [3]

$$\Omega = \left\{ \left\{ \Lambda^i \right\}_{i=1,\ldots,N}, I \right\} \tag{2}$$

którą stanowi skończony zbiór agentów systemu $\left\{ \Lambda^i \right\}_{i=1,\ldots,N}$ oraz relacja zwana relacją współdziałania o postaci

$$I = \bigcup_{\substack{i,j=1,\ldots,N \\ i \neq j}} I^{ij},$$

przy czym

$$A^i \times A^j \supset I^{ij} \ni (a^i, a^j) : A^i \in \Lambda^i; A^j \in \Lambda^j; A^i, A^j \in \Omega.$$

Relacja I jest symetryczna $(a^i, a^j) \in I \Rightarrow (a^j, a^i) \in I$ i określa potencjalne możliwości współdziałania agentów w obrębie systemu. Współdziałanie następuje, gdy agenci wykonują akcje będące w relacji I. Oczywiście obie akcje nie mogą zostać wykonane niezależnie.

Systemem wieloagentowym MAS (*Multi-Agent System*) w ogólnym pojęciu określamy wszelkie rodzaje systemów złożonych z wielu autonomicznych komponentów (encji). Typowymi wzorcami interakcji w MAS są kooperacja (osiąganie wspólnego celu), koordynacja (organizacja współpracy) oraz negocjacja (osiąganie kompromisu).

2.3. Współbieżny wielopopulacyjny algorytm ewolucyjny

Wielopopulacyjne algorytmy ewolucyjne bazują na niezależnym przetwarzaniu zadanej liczby populacji (lub podpopulacji) i okresowej wymianie (migracji) rozwiązań (osobników) między populacjami. Zabieg ten ma za zadanie ukierunkowanie przeszukiwania przestrzeni przez włączenie do populacji rozwiązań o dobrym przystosowaniu i jednoczesne przeciwdziałanie przedwczesnej zbieżności przez różnicowanie materiału genetycznego. Metoda ta na ogół realizuje przetwarzanie losowo wygenerowanych populacji startowych za pomocą tego samego algorytmu ewolucyjnego. Migracje określonej liczby rozwiązań dokonywane są okresowo według zadanego schematu. Emigracje i imigracje osobników usuwają lub pozostawiają kopię osobnika w populacji bazowej. Wymiana rozwiązań może przebiegać pomiędzy losowo wybranymi podpopulacjami bądź sąsiednimi, na podstawie zdefiniowanej relacji porządku. Inną metodą jest automatyczna migracja osobników, które spełniają określone kryteria (np. poprawa funkcji przystosowania najlepszego rozwiązania w populacji) lub wymuszenie migracji do populacji, w których nie uzyskano określonego celu (np. brak poprawy najlepszego rozwiązania w ciągu pewnej liczby iteracji) albo wyczerpano możliwości dalszej eksploatacji przestrzeni rozwiązań (tzn. problem przedwczesnej zbieżności algorytmu spowodowanej niewielkim zróżnicowaniem materiału genetycznego populacji lub zdominowaniem populacji przez jednego osobnika). Podejście wielopopulacyjne w sposób naturalny doskonale nadaje się realizacji równoległej. Wielowątkowe algorytmy ewolucyjne cechują się dodatkowym nietedeterminizmem czasowym spowodowanym synchronizacją.

3. Koncepcja inteligentnego systemu wieloagentowego bazującego na algorytmie ewolucyjnym

Ze względu na występowanie istotnych elementów cechujących inteligentne systemy wieloagentowe zdecydowaliśmy się na przedstawienie koncepcji algorytmu ewolucyjnego, stosując pojęcia i formalizacje obu dziedzin. MAS są najlepszą platformą do budowy inteligentnych systemów adaptacyjnych, gdyż wykorzystują elementy posiadające zdolność uczenia się (inteligencja, adaptacja) oraz mechanizm komunikacji (wymiana doświadczeń), wspomagający ewolucję systemu w celu lepszego dostosowania do środowiska.

3.1. Implementacja MAS

W pracy [4] obok zalet IMAS (odporność, łatwość współpracy z innymi systemami, możliwość rozwiązywanie problemów zdecentralizowanych), wymienia się następujące kwestie implementacyjne, które należy rozwiązać:

- sformułowanie, dekompozycja i przydział zadań agentom oraz syntetyzowanie wyników;
- określenie sposobu komunikacji agentów (jaką informację przekazywać, kiedy i komu);
- udostępnienie agentom wiedzy o innych agentach (reprezentacja wiedzy, wnioskowanie);
- informowanie agentów o efektach ich interakcji (wnioskowanie, czy osiągnięto zadawalające efekty, sposób koordynacji);
- realizacja przez agentów różnych punktów widzenia (konflikty, łączenie);
- projektowanie rzeczywistego systemu IMAS (platforma, metodologia DAI – *distributed artificial intelligence*, uczenie się agentów, planowanie, podejmowanie decyzji);
- zrównoważenie lokalnych obliczeń oraz komunikacji;
- uniknięcie chaotycznego zachowania systemu jako całości;
- tworzenie przez agentów grup, koalicji, prowadzenie negocjacji i osiąganie porozumienia;
- formalna specyfikacja systemu, interakcji agentów i jej weryfikacja.

3.2. Inteligentny agent programowy

Koncepcja inteligentnego agenta programowego bazuje na algorytmie ewolucyjnym. Agentem jest algorytm (jeden z wielu możliwych), który wykorzystując wybrane przez siebie elementy konstrukcyjne i dobierając wartości parametrów stara się poprawić swoją efektywność. Stosowane przez nas podstawowe pojęcia dotyczące opisu EA i jego komponentów można znaleźć w pracy [5] oraz [6]. Jako podstawowy schemat algorytmu przyjęto schemat z zastępowaniem pokoleń (*generational replacement*), w którym przejście do następnego pokolenia powoduje automatyczne zastąpienie rozwiązań – rodziców przez ich potomków. Na EA składają się następujące kroki:

Krok 1. Utworzenie początkowej populacji osobników – generacja losowa.
Krok 2. Ocena przystosowania osobników w populacji – obliczenie funkcji oceny.
Krok 3. Selekcja rozwiązań.
Krok 4. Zastosowanie operatorów genetycznych – krzyżowania i mutacji.
Krok 5. Utworzenie nowej populacji.
Krok 6. Sprawdzenie kryterium zatrzymania – wyprowadzenie najlepszego rozwiązania *(STOP)* lub skok do **kroku 2**.

Implementacja EA wymaga określenia komponentów, z których zostanie on „złożony" oraz wyboru, z wielu możliwych, koncepcji ich realizacji. Należą do nich:

- reprezentacja rozwiązania – kodowanie,
- koncepcja populacji,
- funkcja oceny,
- mechanizm selekcji,
- operatory genetyczne,
- określenie wartości istotnych parametrów.

Dla rozpatrywanego zagadnienia testowego (komiwojażera TSP – *traveling salesman problem*), jako typowego zagadnienia permutacyjnego przyjęto kodowanie całkowitoliczbowe (rzeczywiste), ze względu na prostotę i łatwiejsze, niż w przypadku innego kodowania utrzymanie pewnych własności strukturalnych rozwiązań. Jako funkcję oceny przystosowania przyjęto funkcję celu dla TSP.

Aby uniknąć problemu ze zbieżnością klasycznego podejścia zastępowania pokoleń, zastosowano w populacji elitę, tzn. część rozwiązań populacji przepisuje się do następnej generacji, pozwalając konkurować rozwiązaniom – rodzicom z ich potomstwem.

Faza selekcji jest realizowana w zasadzie przez dwa algorytmy: selekcji oraz wyboru (*sampling*). Pierwszy przyporządkowuje każdemu osobnikowi rzeczywistą liczbę, będącą wartością oczekiwaną liczby potomstwa. Następnie losowane są z prawdopodobieństwem selekcji rozwiązania, w ilości równej liczbie osobników, które mają zostać wymienione w populacji. Do stosowanych u nas koncepcji algorytmów selekcji należą:

- selekcja proporcjonalna do przystosowania,
- selekcja bazująca na rankingu,
- selekcja turniejowa.

Dzięki wykorzystaniu różnych strategii selekcji umożliwiono równoczesne stosowanie następujących podejść:

- selekcji dynamicznej lub statycznej,
- selekcji wygaszającej (*extinctive*) lub zachowującej (*preservative*),
- selekcji czystej (*pure*) lub elitarnej,
- selekcji deterministycznej lub losowej.

Operatory genetyczne specjalizowane dla zagadnienia TSP sprowadzają się do znanych dla zagadnień permutacyjnych operatorów krzyżowania: PMX, OX, CX, dwóch operatorów mutacji oraz dodatkowo operatora optymalizacji lokalnej – poprawiającego zbieżność algorytmu.

Istotnymi parametrami algorytmu ewolucyjnego są wielkość populacji, wielkość elity, prawdopodobieństwa stosowania operatorów, całkowita liczba iteracji algorytmu.

Zbiorem akcji agenta są wszelkie działania dotyczące zmiany struktury algorytmu ewolucyjnego, wartości parametrów i strategii działania. Przykładami akcji są:

- realizacja iteracji EA (np. na zbiorze aktualnych / częściowo / w całości zmienionych rozwiązań);
- dodanie lub usunięcie komponentu algorytmu (np. mechanizm selekcji, operator, elita);
- zmiana intensywności użycia komponentu algorytmu (np. krotność, prawdopodobieństwo);
- zmiana wartości parametrów (np. wielkość populacji, elity);
- zmiana konstrukcji komponentu (np. sposobu traktowania elity, powtórzenia osobników);
- zmiana strategii współdziałania agenta (np. komunikacja, interpretacja wiedzy).

Stan agenta jest określony przez aktualny zbiór rozwiązań optymalizowanego problemu, aktualnie realizowany algorytm (jego elementy i wartości parametrów) oraz przyjętą strategię współdziałania. Strategia współdziałania agenta definiuje warunki, częstość i sposób interakcji. Jest ona definiowana przez kilka elementów, których współwystępowanie

w różnych konfiguracjach tworzy liczną przestrzeń możliwych strategii. Uwzględnianymi elementami są:

- warunek podjęcia interakcji (np. co zadaną liczbę iteracji, w przypadku wystąpienia określonego zdarzenia – poprawa najlepszego rozwiązania, pogorszenie średniej wartości funkcji celu populacji o zadaną wartość progową itp.);
- kierunek przepływu informacji;
- ziarnistość informacji (wiedza szczegółowa, uogólniona, ogólna);
- sposób wnioskowania na podstawie zdobytej wiedzy.

Funkcja przejścia określa wszystkie możliwe przejścia między stanami (w wyniku wykonania akcji) dla agenta realizującego aktualną strategię, czyli definiuje zbiór stanów osiągalnych z aktualnego stanu agenta.

3.3. Interakcja i komunikacja w systemie wieloagentowym

Interakcja jako problem związany z realizacją systemu MAS jest pewnym rodzajem kolektywnej akcji, w której poszczególni agenci podejmują indywidualne akcje na podstawie działań lub wiedzy innych agentów. Z racji współdziałania ukierunkowanego na realizację celu (zwiększenie efektywności algorytmu) interakcja ma celu zdobycie i późniejsze wykorzystanie rozproszonej wiedzy.

Komunikacja asynchroniczna odbywa się z pomocą wspólnej bazy wiedzy (struktury tablicowe), do której agenci przekazują opis stanu (zewnętrzny) oraz pobierają wiedzę (opis stanu innych agentów). Na jej podstawie przeprowadzają wnioskowanie i podejmują akcje zgodnie z aktualną strategią. Opis stanu agenta, wewnętrzny i zewnętrzny, różnią się między sobą liczbą parametrów, w celu ograniczenia komunikacji (np. zamiast wszystkich rozwiązań populacji tylko najlepsze, w przypadku braku sukcesu – wartość funkcji oceny).

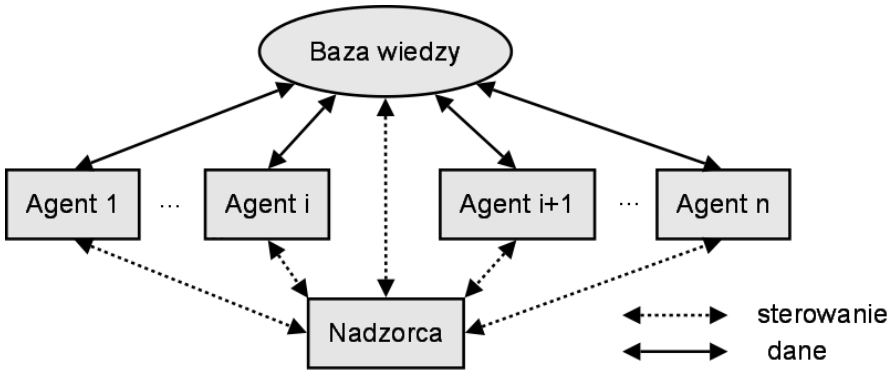
Organizacja społeczności agentów oraz ich współdziałania z perspektywy systemu:

- Agenci realizujący wspólne cele (sterowanie przez strategie kooperacji, mające za zadanie zwiększenie efektywności kolektywnej realizacji celu).
- Ocena realizacji celu agenta dokonywana jest przez pryzmat efektywności realizowanego EA (np. jakość najlepszego uzyskanego rozwiązania, liczba wykonanych iteracji) oraz dodatkowych parametrów charakteryzujących własności populacji (np. różnorodność, perspektywiczność – średnia wartość funkcji celu dla populacji, wartość oczekiwana funkcji celu rozwiązań częściowo ustalonych).
- Dekompozycja zadania na podproblemy i przydział (utworzenie/usunięcie/modyfikacja agentów realizujących strategie odpowiadające różnym perspektywom widzenia problemu – np. próba zwiększenia efektywności EA przez ukierunkowane zmiany wartości wybranego parametru, powielanie strategii wygrywającej i modyfikacja przegrywającej – różnicowanie).
- Model współpracy funkcyjnej dokładnej (*functionally accurate cooperation*) – iteracyjna wymiana wyników częściowych wyłania agenta (agentów) posiadającego informacje i wiedzę (także negatywną), która pozwala dokonać postępu w realizacji zadania nadrzędnego.
- Koordynacja współpracy agentów.

4. Implementacja algorytmu

W ramach przeprowadzonych prac nad implementacją przedstawionej koncepcji systemu agentowego uzyskano rozbudowany EA, w którym wykorzystując parametry konfiguracyjne można zrealizować wiele odmiennych podejść dotyczących przetwarzania populacji (koncepcja pokolenia, populacji, elity), selekcji oraz operatorów genetycznych (początkowo pow. 1000 możliwych wersji).

Na podstawie tego algorytmu zaimplementowano agentów jako zadania (w języku Ada), definiując struktury danych opisujących ich stan (zbiór rozwiązań, elementów EA, wartości parametrów algorytmu i parametry opisujące strategię) oraz architekturę systemu, na który składają się: agenci, agent nadzorca, baza wiedzy, protokoły komunikacji przy wymianie danych i w sterowaniu – (rys. 1).



Rys. 1. Schemat sytemu wieloagentowego

Zadaniem agenta nadzorcy jest zarządzanie systemem MAS od momentu jego zainicjowania, poprzez jego właściwą pracę, do poprawnego zakończenia. Faza zainicjowania systemu polega na stworzeniu zadań będących agentami i przesłaniu do nich wartości początkowych (dotyczących instancji problemu, algorytmu i określających strategię działania). W trakcie właściwej pracy nadzorca zajmuje się obsługą bazy wiedzy (np. wstawianie i usuwanie danych, porządkowanie, wnioskowanie) oraz zarządza tworzeniem i usuwaniem agentów (według przypisanych scenariuszy). Faza zakończenia inicjowana jest przez użytkownika systemu lub wynika z warunku stopu zdefiniowanego w nadzorcy.

Baza wiedzy jest zasobem dzielonym (wspólną strukturą danych), z której mogą korzystać współbieżnie zadania będące agentami. W celu zapewnienia bezpiecznego dostępu (z wzajemnym wykluczaniem) do informacji zawartej w bazie wiedzy jest ona zrealizowana jako obiekt chroniony (jednostka programowa języka Ada). Taki sposób dostępu do bazy realizuje asynchroniczną komunikację pomiędzy agentami (także z nadzorcą).

W przypadku systemów rozproszonych lub dużej liczby agentów istnieje możliwość pewnej decentralizacji działania nadzorcy przez przez stworzenie systemu hierarchicznego z dwoma poziomami nadzorującymi. Przykładowo niższy poziom zawiera *m* nadzorców-

-pośredników (gdzie *m* jest zależne od liczby węzłów systemu rozproszonego), do których przypisani są według określonych kryteriów agenci. Pracę nadzorców-pośredników koordynuje jeden nadzorca pozomu wyższego.

4.1. Model zagadnienia testowego

Jako zadanie testowe przyjęto standardowe zagadnienie komiwojażera (TSP), które należy do klasy problemów permutacyjnych NP-trudnych i jest najbardziej popularnym zagadnieniem optymalizacji dyskretnej. Sprowadza się ono do znalezienia najkrótszego cyklu Hamiltona w spójnym grafie o *n* wierzchołkach. Model matematyczny w ujęciu permutacyjnym możemy zdefiniować następująco:

Dany jest zbiór $N = \{1, ..., n\}$ oraz macierz $(n \times n)$-wymiarowa $A = [a_{i,k}]$. W terminologii teorii grafów: zbiór N jest zbiorem wierzchołków spójnego, ważonego grafu, A macierzą wag krawędzi lub łuków (macierzą kosztów lub odległości), natomiast $\pi(i) \in N$, $i = 1, ..., n$ określa kolejność odwiedzanych wierzchołków (numer i-tego w kolejności wierzchołka).

Należy znaleźć permutację $\pi = (\pi(1), ..., \pi(n))$ elementów zbioru N, która minimalizuje funkcję celu $\phi(\pi)$ o następującej postaci

$$\phi(\pi) = \sum_{i=1}^{n-1} a_{\pi(i)\pi(i+1)} + a_{\pi(n)\pi(1)} \tag{3}$$

Funkcja celu $\phi(\pi)$, $\pi \in \Pi$, określa globalny koszt zamkniętej drogi komiwojażera, natomiast Π jest zbiorem permutacji zbioru N.

Dla zagadnienia TSP zadania testowe zaczerpnięto z bibliotek:

- http://www.tsp.gatech.edu/data/index.html
- http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

4.2. Wybór języka programowania

Biorąc pod uwagę ukierunkowanie na programowanie systemów współbieżnych, implementacje zrealizowano w języku Ada [7]. Jest on uniwersalnym językiem programowania, odznaczającym się wysokim stopniem standaryzacji. Oprogramowanie w wersji źródłowej napisane w tym języku jest łatwo przenaszalne między różnymi platformami np. Windows, Linux, Mac OS X. Aktualna wersja języka Ada 2005 jest standardem międzynarodowym ISO 8652:2007 [8].

Do istotnych zalet języka Ada należą:

- konstrukcje współbieżne w składni języka – łatwość realizacji oprogramowania współbieżnego;
- rozszerzenie w standardzie języka pozwalające na programowanie rozproszone;
- rozwinięte funkcje w zakresie obsługi czasu;
- możliwość zastosowania jako języka projektowania i zapisu algorytmów;
- uniwersalność, duża czytelność kodu, udogodnienia do pracy grupowej;
- możliwość wielokrotnego wykorzystania wcześniej napisanego kodu (*reuse*);

– dobrze określone zasady łączenia z kodem w innych językach programowania np. Fortran, C/C++;
– duży wybór reprezentacji typów dziesiętnych i dostępność operacji arytmetycznych o zwiększonej precyzji.

5. Podsumowanie

Implementacja efektywnie działającego systemy wieloagentowego jest ambitnym wyzwaniem, które nie doczekało się dotychczas zbyt wielu znaczących realizacji [2]. Przyjęcie przez autorów przyrostowego sposobu tworzenia oprogramowania polega na implementacji podstawowej struktury systemu MAS (zawierającej bazową wersję algorytmu ewolucyjnego) i stopniowym wzbogacaniu jej o kolejne elementy takie jak: nowe komponenty algorytmu, strategie postępowania nadzorcy i agentów, sposoby wnioskowania.

Prace, dzięki wykorzystaniu podejścia wielowątkowego, ukierunkowano na wykorzystania procesorów wielordzeniowych, a w przyszłości systemów rozproszonych. Dlatego istotne jest zmniejszenie kosztów komunikacji pomiędzy współbieżnymi wątkami – realizowane dzięki rozproszonej inteligencji i autonomii agentów. Zaproponowana koncepcja systemu MAS jest niezależna od realizacji sprzętowej, w tym liczby procesorów jak i docelowego systemu operacyjnego.

Zastosowane tu podejście wydaje się być skuteczną alternatywą doboru parametrów algorytmu w oparciu o wielowarstwowe heurystyki oraz klasycznych, wielopopulacyjnych algorytmów ewolucyjnych. Dzięki dużej elastyczności proponowanego rozwiązania wiele różnych, opisywanych w literaturze współbieżnych algorytmów ewolucyjnych – bazując na opisanej strukturze podstawowej i modyfikując jedynie parametry nadzorcy, agentów lub bazy danych.

Literatura

[1] Kadłuczka P.: *Hybrydowy algorytm ewolucyjny w optymalizacji uogólnionego zagadnienia podziału grafu.* Kraków, AGH 2003 (Praca doktorska)
[2] Paprzycki M.: *Agenci programowi jako metodologia tworzenia oprogramowania.* e-Informatyka.pl, 2005, http://www.e-informatyka.pl/article/show/422
[3] Dobrowolski G., Nawarecki E.: *Sytuacje kryzysowe w systemach agentowych.* Półrocznik AGH Automatyka, t. 9, z. 1–2, 2005, 57–66
[4] Weiss G., *Multiagent Systems: A modern approach to distributed artificial intelligence.* 1999.
[5] Goldberg D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, 1989; tłum. na jęz. polski: Algorytmy genetyczne i ich zastosowania. Warszawa, WNT 1995
[6] Michalewicz Z.: *Genetic Algorithms + Data Strructures = Evolution Programs*. Springer Verlag, 1995; tłum. na jęz. polski: Algorytmy genetyczne + struktury danych = programy ewolucyjne. Warszawa, WNT 1996
[7] Huzar Z., Fryźlewicz Z., Dubielewicz I., Hnatkowska B., Waniczek J.: *Ada 95*, Helion, 1998
[8] *Ada Reference Manual*. ISO/IEC 8652:2007(E) Ed. 3