Computer Science. Tom 1. 1999 681.3.06:519.686

*Krzysztof Cetnarowicz**

PROBLEM INTEGRALNOŚCI FUNKCJONALNEJ PEWNEJ KLASY SYSTEMÓW WIELOAGENTOWYCH

1. Problem integralności systemów wieloagentowych

Powstające w latach osiemdziesiątych zainteresowanie systemami zdecentralizowanymi doprowadziło do opracowania systemów wieloagentowych opartych na koncepcji autonomicznego agenta.

Systemy wieloagentowe posiadają szereg zalet i otwierają wiele nowych możliwości w tworzeniu systemów informatycznych, jednak wiele problemów związanych z działaniem tych systemów pozostaje do rozwiązania. Do takich problemów należy integralność funkcjonalna systemów wieloagentowych.

Integralność funkcjonalną systemu wieloagentowego można zdefiniować, najogólniej mówiąc, jako zachowanie podstawowych funkcji systemu w czasie jego funkcjonowania. Można ją analizować z punktu widzenia różnych funkcji systemu (które powinny pozostać zachowane) jak i z punktu widzenia różnych czynników jakie mogą wpływać na utratę lub zachowanie integralności funkcjonalnej przez system. W pracy zajęto się badaniem integralności funkcjonalnej systemu wieloagentowego w zależności od ilości agentów (ilość globalna i poszczególnych typów). Agenci w czasie pracy systemu generują agentów tego samego lub innego typu, w zależności od swoich możliwości i potrzeb systemu. Agent mając możliwość postrzegania środowiska w swoim otoczeniu nie ma bezpośredniego dostępu do pewnych globalnych informacji w systemie. Powoduje to, że w wielu przypadkach dany agent generując nowego agenta nie może uwzględnić wszystkich istotnych, z punktu widzenia integralności funkcjonalnej, systemów czynników.

Doprowadza to w konsekwencji do nadmiernej liczby agentów (zablokowanie systemu), lub zbyt małej, aż do zupełnego braku agentów danego typu (zanik funkcji systemu realizowanej przez agentów danego typu).

* Katedra Informatyki, Akademia Górniczo-Hutnicza, Kraków

Autorzy analizujący zachowanie systemów wieloagentowych [8] zwracają uwagę na to, że poza licznymi zaletami jakie mogą posiadać systemy zdecentralizowane mogą one mieć też istotną wadę. Polega ona na tym, że elementy systemu działając w sposób zdecentralizowany zbyt często będą podejmować inicjatywę komunikacji z innymi elementami systemu. Spowoduje to dużą ilość transmisji często zbędnych, a w konsekwencji przeciążenie systemu nadmiernymi transmisjami i w następstwie tego spadek wydajności systemu, a nawet jego zablokowanie. Rozważania przeprowadzono w oparciu o badanie symulacyjne pewnej klasy systemów wieloagentowych.

2. Wieloagentowy system równoważący rozproszenie zasobów w postaci zadań w strukturach wieloprocesorowych

2.1. Budowa wieloagentowego systemu rozpraszania zadań

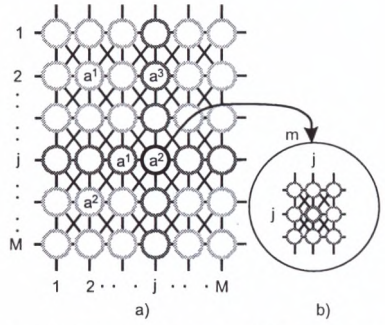
Do realizacji systemu wybrano, należące do bardziej uniwersalnych środowisk, w których mogą działać agenci w systemach wieloagentowych – środowisko grafopodobne.

Środowisko grafopodobne jest oparte na koncepcji grafu niezorientowanego, składającego się z następujących elementów:

- węzłów, w których mogą przebywać agenci i w których mogą znajdować się zasoby. Agenci przebywający w danym węźle mogą się wzajemnie postrzegać i kontaktować, jak też wykorzystywać znajdujące się w danym węźle zasoby według ustalonych reguł;
- krawędzie, za pomocą których agenci mogą przemieszczać się pomiędzy węzłami. Agenci mogą też posiadać zdolność obserwacji pewnych własności węzłów sąsiednich tzn. takich, które są połączone krawędzią z węzłem, w którym dany agent przebywa.

Węzłom nadane są odpowiednie cechy, które określają odpowiednie reguły według jakich agenci, przebywający w danym węźle, mogą się zachowywać i korzystać z zasobów. Agent znajdując się w danym węźle tworzy w swojej świadomości model środowiska, a dokładniej pewnej jego części którą dostrzega (rys. 1).

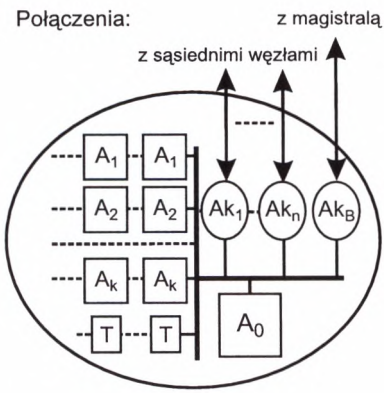
Zasoby znajdujące się w węzłach są z nieprzewidywalną (np. przypadkową) wydajnością produkowane lub pochłaniane. Zadaniem krążących w systemie agentów jest takie przesyłanie (inicjowanie transmisji) zasobu między węzłami, aby rozkład zasobu w węzłach był jak najbardziej zbliżony do zadanego.



Rys. 1. Schemat struktury wieloprocesorowej jako środowiska systemu wieloagentowego i modelu części tego środowiska powstającego w świadomości agenta

Realizacja systemu oparta została na modelu środowiska w postaci struktury wieloprocesorowej złożonej z 400 procesorów połączonych w formie torusa. Każdy z procesorów posiada własną pamięć i może kontaktować się z ośmioma sąsiednimi procesorami za pomocą niezależnych kanałów oraz z dowolnym procesorem struktury za pośrednictwem

magistrali. Z magistrali może w danej chwili korzystać tylko jedna para procesorów (rys. 2). Struktura ma wykonać zadanie składające się z zadań częściowych, których wzajemna kolejność liczenia jest w chwili początkowej nieznana i jest określana (np. losowo) w miarę postępu obliczeń. Rola modułu rozpraszającego, zrealizowanego jako system wieloagentowy, polega na takim rozłożeniu zadań pomiędzy procesory struktury, aby wszystkie zadania częściowe zostały policzone w jak najkrótszym czasie (tzw. *load balancing problem*).



Rys. 2. Schemat węzła grafopodobnej struktury środowiska systemu wieloagentowego

Rozpraszanie za pomocą przesyłania zadań częściowych w strukturze można zrealizować:

- poprzez połączenia z sąsiednimi procesorami (nie zapewnia to zbyt wysokiej efektywności obliczeń);
- za pośrednictwem magistrali (w tym wypadku trzeba każdorazowo ustalić nadawcę i odbiorcę – dwa odległe od siebie w strukturze procesory).

Do realizacji tego celu zastosowano autonomicznych agentów, przemieszczających się w strukturze między procesorami i poszukujących procesorów mających nadmiar zadań częściowych oraz procesorów potrzebujących takich zadań. W szczególności wprowadzono trzy klasy (rodzaje) agentów:

1. Klasa agentów rezydentnych (a^0). Agent rezydentny jest na stałe związany z jednym procesorem, jest wykonywany wyłącznie na tym procesorze i nie może przemieszczać się w strukturze. Jest on generowany dla danego procesora w jednym egzemplarzu i nie może powielać się, ale może generować inne rodzaje agentów.

   Do zadań agenta typu a^0 należy:

   - generacja agentów typu a^1 i a^2 (agent a^0 jest jedynym źródłem tych agentów);
   - transmisja zadań między procesorem, którego dany agent typu a jest rezydentem i procesorami sąsiednimi.

   W pierwszym etapie każdy procesor wyposażony jest w rezydentnego agenta (agent a), którego zadaniem jest poszukiwanie wolnych zadań u sąsiednich procesorów. Poszukiwanie to jest podejmowane w momencie gdy kolejka zadań do wykonania na danym procesorze kończy się.
2. Agenci klasy a^1 mają za zadanie wymianę informacji pomiędzy odległymi (nie będącymi sąsiednimi) procesorami za pomocą magistrali. Agent a nie mogąc uzyskać wolnych zadań u sąsiednich procesorów generuje agentów (a^1), którzy przemieszczając się w strukturze, poszukują zadań dla danego procesora.
3. Dopełnieniem koncepcji agenta a^1 poszukującego wolnych zadań jest stworzenie podobnego agenta a, poszukującego bezczynnego procesora na zlecenie procesora (a dokładniej rezydującego w nim agenta a^0) posiadającego nadmiar zadań.

Autonomiczni agenci mogą być ponadto wykorzystani do pozyskiwania innych informacji umożliwiających poprawę efektywności pracy systemu.

Przedstawiony system dynamicznego rozpraszania zadań w strukturach wieloprocesorowych będący szczególnym przypadkiem problemu dynamicznego rozdziału zasobów, którego bardziej szczegółowy opis można znaleźć w pracach: [2], [3], [4], [5] posłużył jako problem testowy do badania opracowanych koncepcji systemów wieloagentowych i ich własności. Badania prowadzone były na symulacyjnym modelu opisanego systemu wieloprocesorowego.

Jako miarę efektywności (jakości) obliczeń przyjęto wskaźnik E (*efficiency*), w postaci

$$E_f = \frac{T_c}{n * T_r} * 100\% \tag{1}$$

gdzie:

- T_c – czas liczenia wszystkich zadań częściowych na jednym procesorze,
- T_r – czas rzeczywisty uzyskany przy liczeniu wszystkich zadań w strukturze wieloprocesorowej,
- n – liczba procesorów w strukturze.

Jako miarę nierównomierności rozłożenia zadań pomiędzy procesorami struktury przyjęto wskaźnik nierównomierności długości kolejki zadań W_q zdefiniowany następująco:

$$W_q = \frac{n * \max_{1 \le i \le n, 1 \le j \le m} (Nt_{i,j})}{\sum_{i=1,j=1}^{i=n,j=m} (Nt_{i,j})} \tag{2}$$

gdzie:

- $Nt_{i,j}$ – ilość zadań w węźle (procesorze) o numerze (i, j) znajdującym się w wierszu i i kolumnie j,
- n – liczba procesorów w strukturze.

2.2. Integralność funkcjonalna systemów wieloagentowych

Badanie własności systemów wieloagentowych pozwoliło określić pewną cechę nazwaną integralnością funkcjonalną. Problem zachowania integralności funkcjonalnej został sformułowany w ostatnich latach i aczkolwiek jest bardzo ważny dla rozwoju systemów wieloagentowych nie ma dotychczas znaczących wyników badania tego problemu.

Pod nazwą integralności funkcjonalnej można rozumieć zdolność systemu wieloagentowego, traktowanego jako całość, do realizacji pewnych funkcji (zwykle, do realizacji których dany system został zbudowany).

Zachowanie integralności funkcjonalnej systemu można rozumieć jako utrzymanie zdolności (lub jej utracenie) do realizacji wspomnianych powyżej funkcji na skutek określonych czynników.

Jednym ze wspomnianych czynników mającym wpływ na zachowanie integralności funkcjonalnej systemu jest istnienie i liczność grup agentów poszczególnych typów (mogących występować w danym systemie).

Jednym z bardzo istotnych przypadków integralności funkcjonalnej systemu jest zapobieganie nadmiernemu (a nawet nieograniczonemu) wzrostowi ilości pewnych grup agentów, a tym samym globalnej liczby agentów.

Opisany problem można zaobserwować badając zachowanie przedstawionego systemu rozpraszania zadań. Przykładowo spadek globalnej liczby zadań w systemie powoduje, że grupa agentów poszukujących zadań (a^1) wzrasta i utrudnia realizację podstawowych funkcji systemu przez pozostałe grupy agentów co prowadzi do utraty integralności funkcjonalnej przez system.

3. Problem nadmiaru agentów w systemach wieloagentowych i sposoby jego rozwiązywania

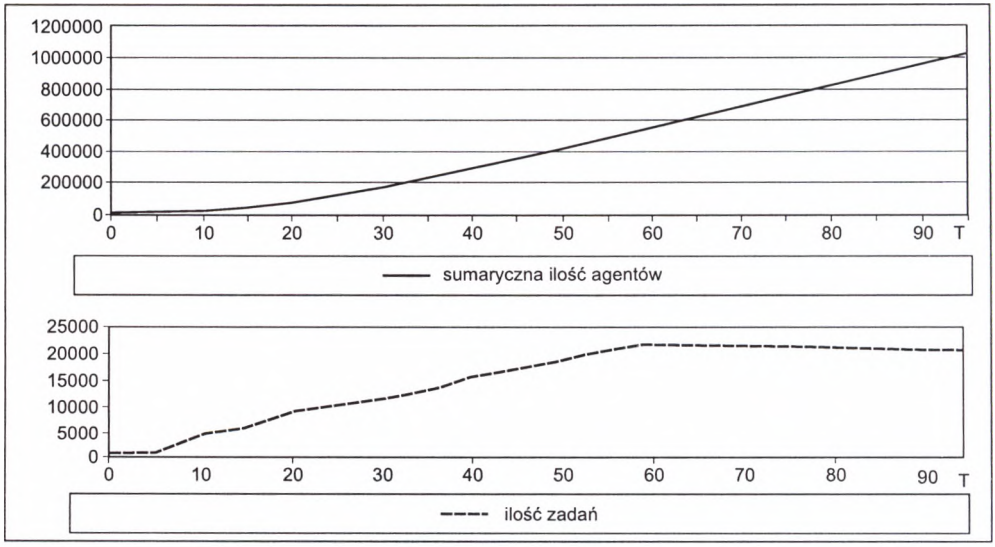
Jak już wspomniano, w systemach agentowych [8] pojawia się problem utraty integralności funkcjonalnej systemu. Polega ona na tym, że w pewnych przypadkach może wystąpić nieograniczony wzrost ilości agentów, co prowadzi do zablokowania systemu. Występujący w dużej ilości agenci, prowadząc wymianę informacji, przemieszczając się itp. absorbują zasoby systemu powodując początkowo spadek wydajności działania systemu, a następnie zablokowanie.

Badania przedstawionego systemu zdecentralizowanego potwierdziły możliwość powstawania przeciążenia systemu przez nadmierną ilość akcji podejmowanych przez zbyt licznych agentów, co prowadzi do nadmiernych transmisji i przetwarzania informacji w systemie, a nawet zablokowania systemu, gdy ilość agentów wzrasta.

Sytuacja taka występuje w przypadku, gdy ilość danego zasobu znacznie spada (lub znacznie wzrasta) niezależnie od działania systemu. W takich przypadkach generowana jest w systemie zbyt duża (ponad potrzeby) ilość agentów, poszukujących zasobu (lub jego odbiorców) co powoduje spadek wydajności systemu i może prowadzić do lawinowego narastania zjawiska i zblokowania systemu. W analizowanym systemie w przypadku braku zasobu powstaje nadmierna ilość agentów typu $g = 1$, a w przypadku nadmiaru zasobu ilość agentów typu $g = 2$. Wyniki symulacyjne potwierdzają przewidywane zjawisko (rys. 3). Sumaryczna ilość agentów typu $g = 1$ *i* $g = 2$ rośnie nieograniczenie i przy ilości agentów równej około jednego miliona powoduje, że praktycznie system zajęty obsługą zleceń agentów nie będąc w stanie przetwarzać zadań. Przy dalszym wzroście ilości agentów następuje zablokowanie dostępu do zasobów systemu i system przestaje pracować.

Dlatego też konieczne jest wprowadzenie mechanizmów pozwalających regulować ilość agentów w systemie. Analizując badany system możemy stwierdzić, że najogólniej rozwiązanie opisanego problemu może być osiągnięte dwiema drogami:

1) Poprzez ograniczenie ilości agentów typu $g = 1$ i $g = 2$ generowanych przez agentów typu $g = 0$. W tym przypadku w momencie wzrostu ilości ponad pewien poziom, następuje włączenie mechanizmów powodujących ogranagentów iczanie ilości wspomnianych agentów.
2) Poprzez stabilizację sumarycznej ilości agentów typu $g = 1$ i $g = 2$ za pomocą koncepcji „wolnego agenta". W tym przypadku ilość agentów (wszystkich lub ustalonych typów) jest ustabilizowana na pewnym poziomie (lub w ustalonych granicach).



Rys. 3. Wykres przedstawiający ilość agentów i ilość zadań w czasie (T) bez ograniczania (stabilizacji) ilości agentów. Liczba agentów rośnie w sposób nieograniczony aż do momentu zablokowania pracy systemu (przy około 1000000 agentów) w systemie

3.1. System wieloagentowy o ograniczonej ilości agentów

Przedstawiony w poprzednim rozdziale (rozdz. 2) system wieloagentowy można wyposażyć w mechanizmy umożliwiające ograniczenie ilości agentów istniejących i działających w danej chwili w systemie.

Ograniczenie to powinno dotyczyć ilości agentów typu $g = 1$ i $g = 2$, natomiast ilość agentów typu $g = 0$ jest zgodnie z przedstawioną definicją systemu stała (po jednym agencie na każdy węzeł struktury).

Agentów typu $g = 1$ i $g = 2$ generują agenci typu $g = 0$ w związku z tym wspomniany mechanizm ograniczający powinien być wbudowany w algorytm działania agenta typu $g = 0$ i jest to zrealizowane za pomocą tzw. „mechanizmu ograniczającego".

Mechanizm ograniczający można wprowadzić przystosowując do tego celu algorytm działania agenta typu $g = 0$, uwzględniając w definicji określającej autonomicznego agenta wspomnianego typu mechanizmy blokujące generację agentów. Proponowane zmiany wprowadzają liczniki umożliwiające śledzenie ilości wygenerowanych agentów typu $g = 1$ i $g = 2$ i porównywanie ich z maksymalnymi ilościami tych agentów, jakie może dany agent typu $g = 0$ wygenerować. Maksymalne dopuszczalne ilości wygenerowanych agentów są określane w momencie generacji każdego agenta typu $g = 0$.

Definicja konfiguracji modeli M^0 Dla rozważanego typu agentów $g = 0$ konfiguracja modeli M^0 wygląda następująco:

$$M^0 = \{m^0 : m^0 = (\mu^0, \alpha^0, \beta^0, \delta^0, \gamma^0)\} \tag{3}$$

$$\alpha^0, \beta^0, \delta^0, \gamma^0 \in \Re \tag{4}$$

$$\mu^0 : II_1 \times II_1 \rightarrow \Re \tag{5}$$

gdzie:

II_{n1}, II_{n2} – zbiory indeksów: $II_n = (-n, -n+1, ..., -1, 0, +1, ..., n-1, n)$.

$$\mu^0 = \{\mu^0(i,j)\} = \{\mu^0_{ij}\} = \begin{pmatrix} \mu^0_{-1,-1} & \mu^0_{-1,0} & \mu^0_{-1,1} \\ \mu^0_{1,-1} & \mu^0_{0,0} & \mu^0_{0,1} \\ \mu^0_{1,-1} & \mu^0_{1,0} & \mu^0_{1,1} \end{pmatrix} \text{gdzie } i, j \in II_1$$

Definicja konfiguracji strategii S^0: Konfiguracja strategii S^0 składa się ze strategii:

$$S^0 = \{sl^0_{i,j}, s2^0_{i,j}, s^0_a, s^0_b\} \quad \text{dla} \quad (i,j) \in II_1 \times II_1 \setminus \{(0,0)\} \tag{6}$$

$sl^0_{i,j}$ – strategia polegająca na pobraniu z węzła $r_{k+i,\, l+j}$ (będącego węzłem sąsiednim dla węzła $r_{k,\, l}$, w którym dany agent typu $g = 0$ się znajduje) pewnej ilości zasobu i przesłanie go do węzła $r_{k,\, l}$. Strategia ta ma następujący wpływ na zmianę modeli w procesie podejmowania decyzji przez agenta:

$$sl^0_{i,j}(\mathrm{m}) = sl^0_{i,j}((\mu^0, \alpha^0, \beta^0, \delta^0, \gamma^0)) = \tag{7}$$

$$= \begin{cases} (\mu^0, \alpha^{0'}, \beta^0, \delta^0, \gamma^0) \text{ gdzie } \alpha^{0'} = \alpha^0 + \mu^0(i,j) \text{ dla } \mu^0(i,j) > 0 \\ (\mu^0, \alpha^0, \beta^{0'}, \delta^0, \gamma^0) \qquad \text{dla } \mu^0(i,j) \le 0 \end{cases}$$

$s2^0_{i,j}$ – strategia polegająca na pobraniu z węzła $r_{k,\, l}$ pewnej ilości zasobu i przesłanie go do węzła $r_{k+i,\ l+j}$ (będącego węzłem sąsiednim dla węzła $r_{k,\, l}$, w którym dany agent typu $g = 0$ się znajduje). Strategia ta ma następujący wpływ na zmianę modeli w procesie podejmowania decyzji przez agenta:

$$s2^0_{ij} = (m) = s2^0_{ij}((\mu^0, \alpha^0, \beta^0, \delta^0, \gamma^0)) = \tag{8}$$

$$= \begin{cases} (\mu^0, \alpha^0, \beta^{0'}, \delta^0, \gamma^0) \text{ gdzie } \beta^{0'} = \beta^0 - \mu^0(i,j) \text{ dla } \mu^0(i,j) < 0 \\ (\mu^0, \alpha^0, \beta^{0'}, \delta^0, \gamma^0) \qquad \text{dla } \mu^0(i,j) \ge 0 \end{cases}$$

s^0_a – strategia polegająca na wygenerowaniu agenta typu $g = 1$ mającego za zadanie odnalezienie węzła, z którego mógłby on pobrać pewną ilość zasobu i przekazać do węzła macierzystego $r_{k,\, l}$. Wygenerowanie agenta następuje jeżeli dany agent typu $g = 0$ nie przekracza maksymalnej dozwolonej liczby agentów typu $g = 1$ jaką wolno mu wygenerować:

$$s_a^0(m) = s_a^0((\mu^0, \alpha^0, \beta^0, \delta^0, \gamma^0)) =$$

$$= \begin{cases} (\mu^0, \alpha^0, \beta^0, \delta^{0'}, \gamma^0) \text{ gdzie } \delta^{0'} = \delta^0 - \mu^0(i,j) \text{ dla } \delta^0 > 0 \\ (\mu^0, \alpha^0, \beta^{0'}, \delta^0, \gamma^0) \qquad \text{dla } \delta^0 \le 0 \end{cases} \tag{9}$$

s_b^0 – strategia polegająca na wygenerowaniu agenta typu $g = 2$ mającego za zadanie odnalezienie węzła, do którego mógłby on przekazać pewną ilość zasobu pobranego z węzła macierzystego $r_{k,l}$. Wygenerowanie agenta następuje jeżeli dany agent typu $g = 0$ nie przekracza maksymalnej dozwolonej liczby agentów typu $g = 2$ jaką wolno mu wygenerować:

$$s_b^0(m) = s_b^0((\mu^0, \alpha^0, \beta^0, \delta^0, \gamma^0)) =$$

$$= \begin{cases} (\mu^0, \alpha^0, \beta^0, \delta^0, \gamma^{0'}) \text{ gdzie } \gamma^{0'} = \gamma^0 - 1 \text{ dla } \gamma^0 > 0 \\ (\mu^0, \alpha^0, \beta^0, \delta^0, \gamma^0) \qquad \text{dla } \gamma^0 \le 0 \end{cases} \tag{10}$$

Definicja konfiguracji celów Q^0: Konfiguracja celów (wskaźników jakości) Q^0 zawiera dwa elementy: q_a^0, q_b^0:

q_a^o – odpowiada przypadkowi, w którym agent typu $g = 0$ stwierdza, że węzeł macierzysty posiada zbyt mało zasobu i należy, jako cel przyjąć uzyskanie wspomnianego zasobu bądź drogą wymiany zasobu z węzłami sąsiednimi bądź poprzez wysłanie agenta (typu 1) poszukującego dawcy. Cel ten określa formuła:

$$q_a^o(m^0, m^{0'}) = q_a^o((\mu^0, \alpha^0, \beta^0, \delta^0, \gamma^0), (\mu^{0'}, \alpha^{0'}, \beta^{0'}, \delta^{0'}, \gamma^{0'})) =$$

$$= \begin{cases} \alpha^{0'} - \alpha^0 \text{ dla } \mu^0(0,0) < 0 \text{ i } \mu^0(0,0) \ge \alpha^0 - \alpha^{0'} \\ \delta^0 - \delta^{0'} \text{ dla } \mu^0(0,0) < 0 \text{ i } \mu^0(0,0) < \alpha^0 - \alpha^{0'} \\ 0 \qquad \text{dla } \mu^0(0,0) = 0 \end{cases} \tag{11}$$

q_b^o – odpowiada przypadkowi, w którym agent typu $g = 0$ stwierdza, że węzeł macierzysty posiada zbyt dużo zasobu i należy jako cel przyjąć uzyskanie wspomnianego zasobu bądź drogą wymiany zasobu z węzłami sąsiednimi, bądź poprzez wysłanie agenta (typu 2) poszukującego odbiorcy. Cel ten określa formuła:

$$q_b^o(m^0, m^{0'}) = q_b^o((\mu^0, \alpha^0, \beta^0, \delta^0, \gamma^0), (\mu^{0'}, \alpha^{0'}, \beta^{0'}, \delta^{0'}, \gamma^{0'})) =$$

$$= \begin{cases} \beta^{0'} - \beta^0 \text{ dla } \mu^0(0,0) > 0 \text{ i } \mu^0(0,0) \le \beta^0 - \beta^{0'} \\ \gamma^0 - \gamma^{0'} \text{ dla } \mu^0(0,0) > 0 \text{ i } \mu^0(0,0) > \beta^0 - \beta^{0'} \\ 0 \qquad \text{dla } \mu^0(0,0) = 0 \end{cases} \tag{12}$$

Definicja funkcji (operatora) obserwacji I^0: Funkcja obserwacji I^0 określa w oparciu o zbiór modeli M^0 i rzeczywiste środowisko, model tego środowiska w świadomości agenta. Stan środowiska V, w którym aktualnie znajduje się agent określa stan węzła, w którym przebywa agent. Zatem środowisko V określone jest przez

$$V = (E, A, C) \tag{13}$$

gdzie:

E = $\{r_{ij}, r\,max_{ij}, r\,min_{ij}\}$ zbiór macierzy reprezentujący przestrzeń grafową, $r_{i,j}$ – aktualna ilość zasobu w węźle (i, j), $r\,max_{i,j}$ $(r\,min_{i,j})$ – maksymalna (minimalna) pożądana ilość zasobu w węźle (i, j),

A = $\varnothing$ zbiór agentów, z którymi dany agent (typu 0) może współdziałać – w rozważanym przypadku zbiór pusty, czyli agent nie współdziała z innymi agentami,

C = $\{(k, l)\}$ jednoelementowy zbiór zawierający parę (k, l), która określa węzeł (wiersz i kolumnę), w którym dany agent typu $g = 0$ znajduje się aktualnie – jest to dla niego węzeł macierzysty.

Zatem funkcja I określa:

$$I^0(M^0, (\{r_{ij}, r\,max_{ij}, r\,min_{ij}\},\varnothing, \{(k, l)\})) = m^0 = (\mu^0, \alpha^0, \beta^0, \delta^0, \gamma^0) \tag{14}$$

gdzie:

$$\mu^0 = \begin{cases} r(k+i,l+j) - r\,max(k+i,l+j) & \text{gdy } r(k+i,l+j) > r\,max\ (k+i,l+j) \\ r(k+i,l+j) - r\,min(k+i,l+j) & \text{gdy } r(k+i,l+j) < r\,min\ (k+i,l+j) \\ 0 & \text{w przeciwnym przypadku} \end{cases}$$

dla $i, j \in II_l$

$\alpha^0 = 0$

$\beta^0 = 0$

δ^0 = ilość agentów typu $g = 1$ jaką dany agent typu $g = 0$ może wygenerować

γ^0 = ilość agentów typu $g = 2$ jaką dany agent typu $g = 0$ może wygenerować.

Definicja funkcji (operatora) realizacji strategii X^0: Funkcja realizacji strategii X^0 jest dla zdefiniowanych strategii określona:

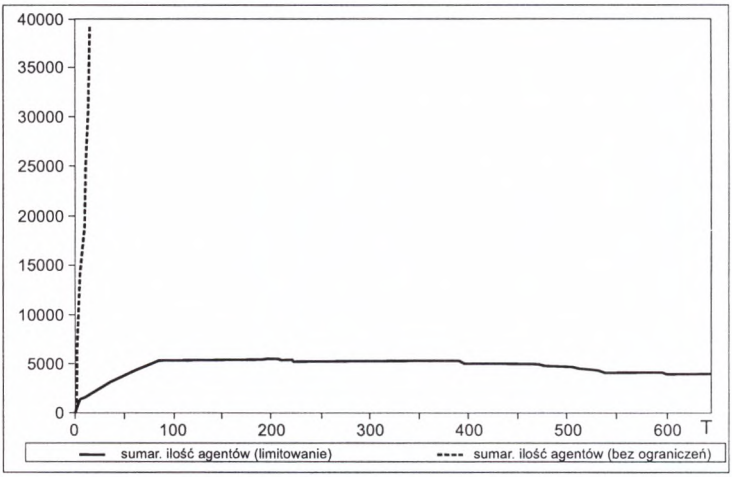
- $X^0(s1^0_{i,j}, (\{r_{ij}, r\,max_{ij}, r\,min_{ij}\},\varnothing, \{(k, l)\}))$ – pobierz z węzła sąsiedniego o współrzędnych $(k + i, l + j)$ ilość $\mathrm{rmax}(k + i, l + j) - r(k + i, l + j)$ zasobu i przekaż go do węzła macierzystego o współrzędnych (k, l);
- $X^0(s2^0_{i,j}, (\{r_{ij}, r\,max_{ij}, r\,min_{ij}\},\varnothing, \{(k, l)\}))$ – pobierz z węzła macierzystego o współrzędnych (k, l). ilość $\mathrm{rmin}(k, l) - r(k, l)$ zasobu i przekaż go do węzła sąsiedniego o współrzędnych $(k + i, l + j)$;

– $X^0(s_a^0, (\{r_{ij}, r\,max_{ij}, r\,min_{ij}\}, \varnothing, \{(k, l)\}))$ – wygeneruj agenta typu $g = 1$ w węźle o współrzędnych (k, l) i przydziel mu ten węzeł jako węzeł macierzysty.
– $X^0(s_b^0, (\{r_{ij}, r\,max_{ij}, r\,min_{ij}\}, \varnothing, \{(k, l)\})$ – wygeneruj agenta typu $g = 2$ w węźle o współrzędnych (k, l) i przydziel mu ten węzeł jako węzeł macierzysty.

W efekcie dany agent typu $g = 0$, a tym samym procesor, z którym dany agent jest związany, może mieć w każdej chwili działania systemu ilość wygenerowanych agentów typu $g = 1$ od zera do wartości ustalonej przez parametr δ i agentów typu $g = 2$ od zera do wartości ustalonej przez parametr γ. Parametry δ i γ są ustalane arbitralnie dla każdego agenta typu $g = 0$ jako parametry konfiguracyjne systemu dobierane na drodze doświadczalnej.

Wyniki badań symulacyjnych wieloagentowego systemu z mechanizmem limitowania ilości agentów. Badania symulacyjne systemu wieloagentowego potwierdzają skuteczność proponowanego rozwiązania, polegającego na określeniu limitu agentów jakich może wygenerować dany procesor, a w szczególności agent typu $g = 0$ związany z danym węzłem.

Na wykresie przedstawionym na rys. 4 sumaryczna ilość agentów pozostaje na stosunkowo niskim poziomie i nie blokuje pracy systemu.



Rys. 4. Wykres przedstawiający sumaryczną ilość agentów w czasie (T). Wariant z limitowaną liczbą agentów i wariant bez ograniczania liczby agentów

Przedstawiony mechanizm ograniczania ilości agentów, aczkolwiek skuteczny, posiada pewne niedoskonałości mające wpływ na optymalną pracę systemu:

– Teoretyczny opis procesu uwzględniający zmieniającą się ilość agentów i zapotrzebowanie systemu na ich generację jest trudny (o ile w ogóle możliwy) i stąd wydaje się, że określenie limitów ilości agentów poszczególnych typów jakie może mieć wygenerowane dany agent typu $g = 0$ (dany węzeł) jest możliwe w praktyce tylko na drodze doświadczalnej. Co więcej, trudne jest nawet określenie przesłanek logicznych, umożliwiających optymalne oszacowanie zapotrzebowania na agentów wspomnianych typów.

– Niekorzystny dla optymalnej pracy systemu jest nie tylko sumaryczny globalny nadmiar ilości agentów, ale również ich nadmiar lokalny powstający w danej części systemu (poszczególnych procesorach), czego proponowane podejście nie rozwiązuje.

3.2. Ograniczenie ilości agentów za pomocą mechanizmu usuwania nadmiarowych agentów

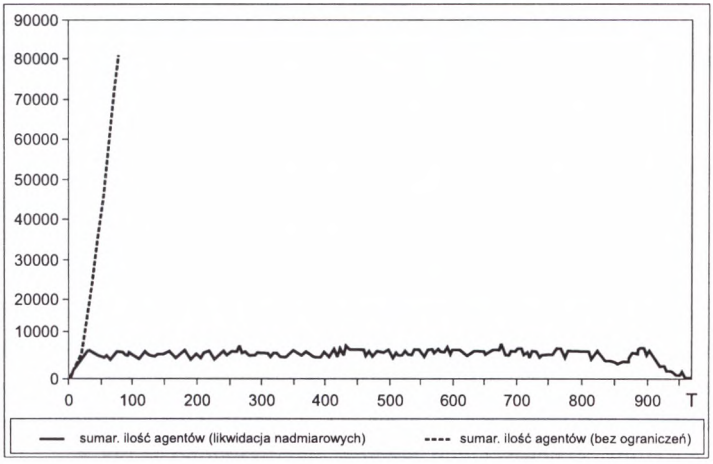
Do regulowania ilości agentów może być wykorzystany mechanizm usuwania (likwidacji, „zabijania") nadmiarowych agentów. Mechanizm ten opiera się na dokładnej identyfikacji skutków, jakie mogą powodować nadmierne ilości agentów w systemie. W oparciu o analizę wspomnianych skutków podejmowane są decyzje o tym, jaką ilość agentów jakiego typu usunąć.

Analizując przedstawiony w rozdziale 2 system, można zauważyć, że odpowiedzialny za pracę procesora jest agent typu $g = 0$ związany z danym procesorem. Agent ten, między innymi, wykonuje operacje:

– przetwarzania zadań,
– realizacji usług dla agentów znajdujących się w danej chwili w danym węźle.

Jeżeli ilość agentów znajdujących się w danym węźle (przechodzących przez dany węzeł) wzrasta, agent typu $g = 0$ musi poświęcić więcej czasu na realizację operacji dla wspomnianych agentów zamiast na liczenie zadań. W takim przypadku wykonywanie zadań w danym procesorze ulega zwolnieniu.

Każdy agent typu $g = 0$ jest rozliczany z efektywności obliczania zadań, a więc wspomniane powyżej zjawisko staje się dla tegoż agenta niekorzystne. Biorąc pod uwagę powyższe rozważania można zauważyć, że to właśnie rezydentni agenci typu $g = 0$ mogą zaobserwować niekorzystne konsekwencje nadmiaru agentów w systemie i podjąć odpowiednie działania zmierzające do ograniczania ich ilości poprzez likwidację nadmiaru (rys. 5).



Rys. 5. Wykres przedstawiający ilość agentów w czasie (T). Wariant z mechanizmem usuwania nadmiarowych agentów porównany z wariantem bez ograniczania liczby agentów

3.3. Ograniczenie ilości agentów za pomocą mechanizmu uzależniającego decyzję generacji nowego agenta od ilości agentów tego samego typu w sąsiadującym środowisku

Analizując problem pojawiania się zbyt wielkiej liczby agentów w systemie, można dotrzeć do źródła problemu. W celu ograniczenia liczby agentów danego typu należy kontrolować i ograniczać ilość agentów generowanych. Zatem należy wprowadzić mechanizmy kontroli procesu generacji agentów w systemie.

Analizując przykład systemu przedstawionego w rozdziale 2 można zauważyć, że agentem odpowiedzialnym za generację agentów wszystkich typów, które mogą być generowane w czasie pracy systemu, są agenci typu $g = 0$. Zatem mechanizm kontroli ilości generowania agentów powinien być wbudowany w algorytm działania agenta tego typu.

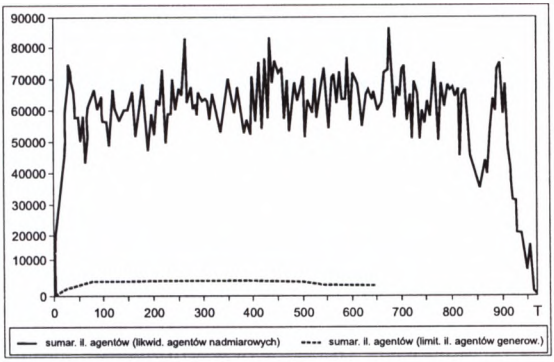
Przyjmijmy dla ustalenia uwagi, że agent typu $g = 0$ ma zamiar wygenerować nowego agenta poszukującego zadań (typu $g = 1$). Schemat postępowania wspomnianego agenta typu $g = 0$, uwzględniający kontrolę ilości wygenerowanych agentów, jest następujący:

- Wspomniany agent typu $g = 0$ obserwuje najbliższe środowisko tzn. węzeł, W którym się znajduje, zwracając uwagę na agentów typu $g = 1$ znajdujących się aktualnie w tym węźle. Są to agenci, którzy przechodzą przez dany węzeł w poszukiwaniu zasobu (zadań). Każdy z tych agentów posiada pewien zasób energii, który ustalany jest przy generacji każdego agenta typu $g = 1$ i pomniejszany w czasie przemieszczania się agenta między węzłami (o ustaloną porcję energii na każde przejście między sąsiednimi węzłami). Ilość energii każdego agenta typu $g = 1$ może być obserwowana przez danego agenta typu $g = 0$.
- Wspomniany agent rezydentny typu $g = 0$ zlicza ilość agentów typu $g = 1$, których poziom energii jest mniejszy od pewnego ustalonego (np. na drodze doświadczalnej) poziomu. Duża ilość takich agentów w danym węźle świadczy o bezskutecznym poszukiwaniu zadań w strukturze przez inne węzły (a dokładniej ich agentów typu $g = 0$), co pozwala z dużym prawdopodobieństwem przyjąć, że w środowisku (strukturze wieloprocesorowej) brakuje poszukiwanego zasobu (wolnych zadań).
- Jeżeli ilość agentów typu $g = 1$ o poziomie energetycznym mniejszym od zadanego poziomu jest większa od pewnej przyjętej ilości (np. ustalonej doświadczalnie), to dany agent nie generuje nowego agenta poszukującego zadań (typu $g = 1$) opierając się na założeniu, że w środowisku brakuje danego zasobu (wolnych zadań).

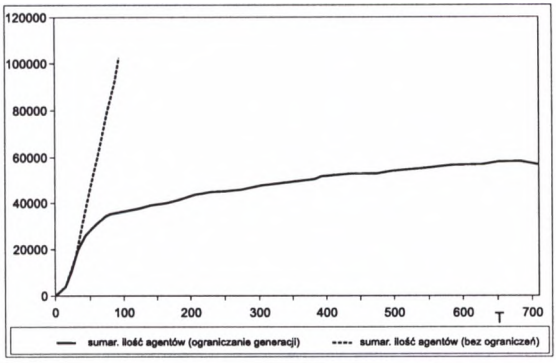
Wyniki praktycznego badania wpływu opisanego mechanizmu powstrzymywania się od generowania nowych agentów na ogólną liczbę agentów w systemie przedstawiono na rys. 6 i 7.

Natomiast wynik współdziałania opisanego mechanizmu ograniczania generacji nowych agentów z algorytmem ograniczania ilości agentów przez limitowanie (dla danego agenta typu $g = 0$) ilości generowanych agentów przedstawiono na rys. 8.

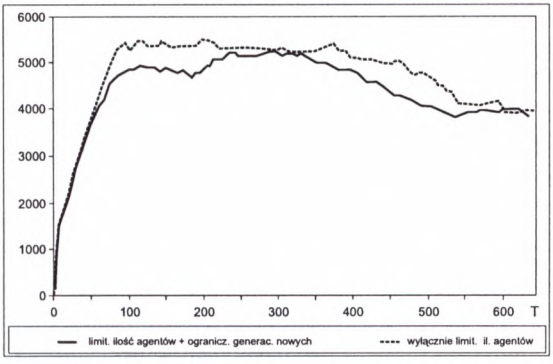
Wyniki potwierdzają przydatność mechanizmu powstrzymywania się od generacji agentów do stabilizacji liczby agentów w systemach wieloagentowych.



Rys. 6. Wykres przedstawiający ilość agentów w czasie (T). Wariant z mechanizmem usuwania nadmiarowych agentów porównany z wariantem w którym ograniczenie ilości agentów jest realizowane poprzez ograniczanie liczby (powstrzymanie się od generacji) generowanych agentów typu $g = 1$ i $g = 2$ przez agenta typu $g = 0$



Rys. 7. Wykres przedstawiający ilość agentów w czasie (T). Wariant z ograniczaniem ilości generowanych agentów (decyzja powstrzymania się od generacji) i wariant bez ograniczeń generacji agentów



Rys. 8. Wykres przedstawiający ilość agentów w czasie (T). Wariant wyłącznie z mechanizmem ograniczania ilości generowanych agentów (powstrzymanie się od generacji) i wariant z limitowaniem uzupełnionym o mechanizm powstrzymywania się od generowania agentów nadmiarowych

4. Koncepcja wolnych agentów i ich zastosowanie do stabilizacji ilości agentów w danej populacji

Wprowadzenie pojęcia wolnego agenta umożliwia skuteczne i stosunkowo proste ograniczenie liczby agentów w danej populacji.

Zastosowanie koncepcji wolnych agentów umożliwia ustalenie sumarycznej ilości agentów w danym systemie, umożliwiając zwiększanie ilości agentów jednego typu kosztem ilości agentów innego typu – w miarę potrzeb związanych z zadaniami wykonywanymi przez system.

Ponadto koncepcja ta umożliwia sprawne przegrupowanie agentów z jednych części środowiska do innych – tam gdzie agenci danego typu są aktualnie potrzebni.

Koncepcja wolnego agenta opiera się na zasadzie, że każdy agent może posiadać zdolność generowania innego agenta takiego samego jak on sam lub innego typu.

Wolny agent jest agentem, który nie wykonuje żadnego zleconego zadania. Przemieszczając się w przestrzeni poszukuje dla siebie zadań do realizacji. Jest to zatem rodzaj „bezrobotnego" agenta poszukującego „pracy". W szczególności działanie wolnego agenta może być przedstawione w następujących punktach:

- dany agent pewnego ustalonego typu podejmuje działania w celu wykonania zleconych zadań;
- dochodzi do sytuacji, w której wspomniany agent nie może kontynuować swojej działalności i ma ulec likwidacji. Może to nastąpić w dwóch przypadkach:
  1) agent dochodzi do wniosku, że dalsza działalność agenta jest niemożliwa lub bezcelowa (np. zadanie zostało wykonane) i wówczas wykonuje operację samolikwidacji,
  2) dalsze działanie agenta jest niemożliwe w wyniku stanu energetycznego agenta i agent jest likwidowany – ginie z braku energii życiowej;
- agent przed wykonaniem operacji likwidacji generuje agenta wolnego;
- agent wolny przemieszczając się w środowisku sprawdza, czy w oparciu o stan środowiska w poszczególnych jego częściach nie ma zapotrzebowania na agenta któregoś z dostępnych w danym systemie typów;
- jeżeli agent wolny stwierdzi, że istnieje zapotrzebowanie na agenta danego typu ($g = 1$ lub $g = 2$) generuje takiego agenta, a sam ulega likwidacji.

Jak już wspomniano koncepcja wprowadzenia wolnych agentów umożliwia ustalenie liczby agentów wszystkich typów w systemie, i dokonywanie jedynie zmian procentowego udziału agentów poszczególnych typów.

4.1. Zastosowanie koncepcji wolnych agentów do stabilizacji ilości agentów w danej populacji w systemie dynamicznego rozdziału zasobów

Zastosowanie koncepcji wolnego agenta w przedstawionym przykładzie dynamicznego rozpraszania zadań (rozdz. 2), wymaga wprowadzenia nowego typu agentów – wolnych agentów (typ $g = 3$) i zmian w budowie i algorytmie działania agentów typu $g = 0$, $g = 1$, $g = 2$.

4.1.1. Podstawowe definicje określające autonomicznego agenta typu $g = 0$

Dla agentów typu $g = 0$ dostosowanie do działania w systemie z wolnymi agentami wymaga wprowadzenia zmian:

- w definicji konfiguracji M^0, S^0, Q^0,
- w definicji operacji I^0, X^0.

Wprowadzane zmiany są określone w oparciu o definicje przedstawione w rozdziale 2, 3.

Definicja konfiguracji modeli M^0: Dla rozważanego typu agentów $g = 0$ konfiguracja modeli M^0 wygląda następująco:

$$M^0 = \{m^0 : m^0 = (\mu^0, \alpha^0, \beta^0, \delta^0, \gamma^0)\} \tag{15}$$

$$\alpha^0, \beta^0, \eta^0 \in \Re, \mu^0 : II_1 \times \mathrm{II}_1 \rightarrow \Re \tag{16}$$

gdzie:

II_{n1}, II_{n2} – zbiory indeksów jak zdefiniowano w rozdziale 2, 3.

W przedstawionym modelu m^0 ($m^0 \in M^0$):

- Macierz μ^0 jest obrazem otoczenia agenta zawierającym wybrane informacje dotyczące sąsiednich węzłów. W rozważanym przykładzie są to informacje o ilości zasobu w tych węzłach. Forma macierzy μ^0 jest taka jak zdefiniowano w rozdziale 2, 3.
- α^0 reprezentuje ilość zasobu zabranego przez agenta typu $g = 0$ z węzła sąsiedniego będącego dawcą zasobu dla danego węzła (podobnie jak w odpowiedniej definicji w rozdziale 2, 3).
- β^0 reprezentuje ilość zasobu przekazanego przez agenta typu $g = 0$ do węzła sąsiedniego, będącego biorcą zasobu z danego węzła (podobnie jak w odpowiedniej definicji w rozdziale 2, 3).
- η^0 reprezentuje ilość agentów typu $g = 1$ i typu $g = 2$ jaką może jeszcze być wygenerowana przez danego agenta typu $g = 0$.

Definicja konfiguracji strategii S^0:

$$S^0 = \{s1^0_{i,j}, s2^0_{i,j}, s^0_a, s^0_b\} \text{dla } (i,j) \in \mathrm{II}_1 \times II_1 \setminus \{(0,0)\} \tag{17}$$

Strategie:

$s1^0_{i,j}$ – strategia polegająca na pobraniu z węzła $r_{k+i,\, l+j}$ (będącego węzłem sąsiednim dla węzła $r_{k,\, l}$, w którym dany agent typu $g = 0$ znajduje się) pewnej ilości zasobu i przesłanie go do węzła $r_{k,l}$,

$s2^0_{i,j}$ – strategia polegająca na pobraniu z węzła $r_{k,l}$ pewnej ilości zasobu i przesłanie go do węzła $r_{k+i,\, l+j}$ (będącego węzłem sąsiednim dla węzła $r_{k,l}$ w którym dany agent typu $g = 0$ znajduje się),

są zdefiniowane identycznie jak w rozdziale 2, 3.

Natomiast definicje pozostałych strategii posiadają pewne zmiany:

s_a^0 – strategia polegająca na wygenerowaniu agenta typu $g = 1$ mającego za zadanie odnalezienie węzła, z którego mógłby on pobrać pewną ilość zasobu i przekazać do węzła macierzystego $r_{k,l}$. Strategia ta ma następujący wpływ na zmianę modeli w procesie podejmowania decyzji przez agenta:

$$s_a^0(m) = s_a^0((\mu^0, \alpha^0, \beta^0, \delta^0, \gamma^0)) = \\ = \begin{cases} (\mu^0, \alpha^0, \beta^0, \eta^{0'}) \text{ gdzie } \eta^{0'} = \eta^0 - 1 & \text{dla } \mu^0(0,0) < 0 \\ (\mu^0, \alpha^0, \beta^0, \eta^0) & \text{dla } \mu^0(0,0) \geq 0 \end{cases} \tag{18}$$

s_b^0 – strategia polegająca na wygenerowaniu agenta typu $g = 2$ mającego za zadanie odnalezienie węzła, do którego mógłby on przekazać pewną ilość zasobu pobranego z węzła macierzystego $r_{k,l}$. Strategia ta ma następujący wpływ na zmianę modeli w procesie podejmowania decyzji przez agenta:

$$s_b^0(m) = s_b^0((\mu^0, \alpha^0, \beta^0, \delta^0, \gamma^0)) = \\ = \begin{cases} (\mu^0, \alpha^0, \beta^0, \eta^{0'}) \text{ gdzie } \eta^{0'} = \eta^0 - 1 & \text{dla } \mu^0(0,0) > 0 \\ (\mu^0, \alpha^0, \beta^0, \eta^0) & \text{dla } \mu^0(0,0) \leq 0 \end{cases} \tag{19}$$

Definicja konfiguracji celów Q^0: Podobnie jak w definicji przedstawionej w rozdziale 2, 3 konfiguracja celów Q^0 zawiera dwa elementy, z których każdy określa inny cel:

q_a^0 – odpowiada przypadkowi, w którym agent typu $g = 0$ stwierdza, że węzeł macierzysty posiada zbyt mało zasobu i należy jako cel przyjąć uzyskanie wspomnianego zasobu bądź drogą wymiany zasobu z węzłami sąsiednimi, bądź poprzez wysłanie agenta (typu 1) poszukującego dawcy. Wybór węzła sąsiedniego, o ile odpowiednie węzły sąsiednie istnieją, następuje pod kątem maksymalizacji ilości pobranego lub oddanego zasobu.

Cel ten określa formuła:

$$q_a^o(m^0, m^{0'}) = q_a^o((\mu^0, \alpha^0, \beta^0, \delta^0, \gamma^0), (\mu^{0'}, \alpha^{0'}, \beta^{0'}, \delta^{0'}, \gamma^{0'})) = \\ = \begin{cases} \alpha^{0'} - \alpha^0 & \text{dla } \mu^0(0,0) < 0 \text{ i } \mu^0(0,0) \geq \alpha^0 - \alpha^{0'} \\ \eta^{0'} - \eta^0 & \text{dla } \mu^0(0,0) < 0 \text{ i } \mu^0(0,0) < \alpha^0 - \alpha^{0'} \\ 0 & \text{dla } \mu^0(0,0) \leq 0 \end{cases} \tag{20}$$

q_b^0 – odpowiada przypadkowi, w którym agent typu $g = 0$ stwierdza, że węzeł macierzysty posiada zbyt dużo zasobu i należy jako cel przyjąć uzyskanie wspomnianego zasobu bądź drogą wymiany zasobu z węzłami sąsiednimi, bądź poprzez wysłanie agenta (typu 2) poszukującego odbiorcy. Wybór węzła sąsiedniego, o ile odpowiednie węzły sąsiednie istnieją, następuje pod kątem maksymalizacji ilości pobranego lub oddanego zasobu.

Cel ten określa formuła:

$$q_b^o(m^0,m^{0'}) = q_b^o((\mu^0, \alpha^0, \beta^0, \delta^0, \gamma^0), (\mu^{0'}, \alpha^{0'}, \beta^{0'}, \delta^{0'}, \gamma^{0'})) = \tag{21}$$

$$= \begin{cases} \beta^{0'} - \beta^0 & \text{dla } \mu^0(0,0) > 0 \text{ i } \mu^0(0,0) \leq \beta^{0'} - \beta^0 \\ \eta^{0'} - \eta^0 & \text{dla } \mu^0(0,0) > 0 \text{ i } \mu^0(0,0) > \beta^{0'} - \beta \\ 0 & \text{dla } \mu^0(0,0) \leq 0 \end{cases}$$

Definicja funkcji (operatora) obserwacji I^0: Funkcja obserwacji I^0, podobne jak w rozdziale 2, 3 określa w oparciu o zbiór modeli M^0 i rzeczywiste środowisko model tego środowiska w świadomości agenta. Stan środowiska, w którym aktualnie znajduje się agent określa stan węzła, w którym przebywa agent w postaci

$$V = (E, A, C) \tag{22}$$

gdzie:

E = zbiór macierzy reprezentujacy przestrzen grafowa (rozdz. 3),

A = $\varnothing$ zbiór agentów, z którymi dany agent (typu 0) może współdziałać – w rozważanym przypadku zbiór pusty, czyli agent nie współdziała z innymi agentami i ich po prostu nie zauważa,

C = $\{(k, l)\}$ jednoelementowy zbiór zawierający parę (k, l) która określa węzeł (wiersz i kolumnę), w którym dany agent typu $g = 0$ aktualnie się znajduje – jest to dla niego węzeł macierzysty.

Funkcja I określa:

$$I^0(M^0, (\{r_{ij}, r\,max_{ij}, r\,min_{ij}\}, \varnothing, \{(k, l)\})) = m^0 = (\mu^0, \alpha^0, \beta^0, \eta^0) \tag{23}$$

gdzie:

$\alpha^0 = 0$

$\beta^0 = 0$

η^0 = ilość agentów typu $g = 1$ lub typu $g = 2$ jakie może wygenerować dany agent (typu $g = 0$),

$$\mu^0 = \mu^0\,\{\mu^0(i,j)\} = \begin{pmatrix} \mu^0_{-1,-1} & \mu^0_{-1,0} & \mu^0_{-1,1} \\ \mu^0_{0,-1} & \mu^0_{0,0} & \mu^0_{0,1} \\ \mu^0_{1,-1} & \mu^0_{1,0} & \mu^0_{1,1} \end{pmatrix}$$

gdzie:

$$\mu^0(i,j) = \begin{cases} r(k+i, l+j) - r\,max(k+i, l+j) & \text{gdy } r(k+i, l+j) > r\,max(k+i, l+j) \\ r(k+i, l+j) - r\,min(k+i, l+j) & \text{gdy } r(k+i, l+j) < r\,min(k+i, l+j) \\ 0 & \text{w przeciwnym przypadku} \end{cases}$$

dla $i, j \in II_1$.

Definicja funkcji (operatora) realizacji strategii X^0: Funkcja realizacji strategii X^0 jest dla zdefiniowanych strategii określona:

- $X^0(\ s1^0_{i,j}, (\{r_{ij}, r\ max_{ij}, r\ min_{ij}\}, \varnothing, \{(k, l)\}))$ – pobierz z węzła sąsiedniego o współrzędnych $(k + i, l + j)$ ilość $\text{rmax}(k + i, l + j) - r(k + i, l + j)$ zasobu i przekaż go do węzła macierzystego o współrzędnych (k, l),
- $X^0(\ s2^0_{i,j}, (\{r_{ij}, r\ max_{ij}, r\ min_{ij}\}, \varnothing, \{(k, l)\}))$ – pobierz z węzła macierzystego o współrzędnych (k, l). ilość $rmin(k, l) - r(k, l)$ zasobu i przekaż go do węzła sąsiedniego o współrzędnych $(k + i, l + j)$,
- $X^0(\ s^0_a, (\{r_{ij}, r\ max_{ij}, r\ min_{ij}\}, \varnothing, \{(k, l)\}))$ – wygeneruj agenta typu $g = 1$ w węźle o współrzędnych (k, l) i przydziel mu ten węzeł jako węzeł macierzysty,
- $X^0(\ s^0_b, (\{r_{ij}, r\ max_{ij}, r\ min_{ij}\}, \varnothing, \{(k, l)\}))$ – wygeneruj agenta typu $g = 2$ w węźle o współrzędnych (k, l) i przydziel mu ten węzeł jako węzeł macierzysty.

4.1.2. Podstawowe definicje określające autonomicznego agenta typu $g = 1$

Dla agentów typu $g = 1$:

- konfiguracje M^1, S^1, Q^1,
- operacje I^1,

są określone identycznie jak odpowiednie definicje w rozdziałach 2 i 3. Natomiast definicja operacji X^1 posiada pewne modyfikacje:

Definicja funkcji (operatora) realizacji strategi X^1 Funkcja realizacji strategii X^1 jest dla zdefiniowanych strategii określona:

- $X^1(\ s^1_0, V = (\{r_{ij}, r\ max_{ij}, r\ min_{ij}\}, \varnothing, \{(k, l)\}))$ – pobierz z węzła, w którym się znajdujesz (o współrzędnych (k, l)) ilość $r\ max(k, l) - r(k, l)$ i wyślij go do węzła macierzystego, wygeneruj agenta typu $g = 3$ w danym węźle a następnie dokonaj samolikwidacji agenta generując wolnego agenta,
- $X^1(\ s^1_{i,j}, V = (\{r_{ij}, r\ max_{ij}, r\ min_{ij}\}, \varnothing, \{(k, l)\}))$ – przemieść się do węzła reprezentowanego przez element macierzy r o numerze $k + i, l + j$ (kolumny i wiersza).

Przyjmując opisany algorytm działania agent w oparciu o przedstawione definicje poszukuje węzła, w którym ilość zasobu jest większa od zera i po znalezieniu przesyła znalezioną ilość zasobu do węzła macierzystego. Różnica w porównaniu z algorytmem działania agentów typu $g = 1$, opisanych w rozdziale 2 polega na tym, że agent ten przed wykonaniem operacji samolikwidacji generuje agenta wolnego (agenta typu $g = 3$).

4.1.3. Podstawowe definicje określające autonomicznego agenta typu $g = 2$

Dla agentów typu $g = 2$:

- konfiguracje M^2, S^2, Q^2,
- operacje I^2,

są określone tak jak to przedstawiono w rozdziałach 2 i 3. Natomiast definicja operacji X^2 posiada inną postać.

Definicja funkcji (operatora) realizacji strategii X^2: Funkcja realizacji strategii X^2 jest dla zdefiniowanych strategii określona:

- $X^2(s_0^2, V = (\{r_{ij}, rmax_{ij}, rmin_{ij}\}, \varnothing, \{(k, l)\}))$ – przekaż do węzła, w którym się znajdujesz (o współrzędnych (k, l)) ilość r^2 pobranego (przesłanego) z węzła macierzystego, a następnie dokonaj samolikwidacji agenta generując wolnego agenta,
- $X^2(s_{i,j}^2, V = (\{r_{ij}, rmax_{ij}, rmin_{ij}\}, \varnothing, \{(k, l)\}))$ – przemieść się do węzła reprezentowanego przez element macierzy r o numerze $k + i$, $l + j$ (kolumny i wiersza).

Przyjmując opisany algorytm działania agent w oparciu o przedstawione definicje agent poszukuje węzła, w którym ilość zasobu jest większa od zera i po znalezieniu przesyła znalezioną ilość zasobu do węzła macierzystego. Różnica w porównaniu z algorytmem działania agentów typu $g = 2$ opisanych w rozdziale 2, 3 polega na tym, że agent ten przed wykonaniem operacji samolikwidacji, generuje agenta wolnego (agenta typu $g = 3$).

4.1.4. Podstawowe definicje określające autonomicznego agenta typu g = 3

Dla agentów typu $g = 3$ określone zostaną:

- Konfiguracje M^3, S^3, Q^3,
- operacje I^3, X^3.

Definicja konfiguracji modeli M^3: Dla rozważanego typu agentów $g = 1$ konfiguracja modeli M^3 wygląda następująco:

$$M^3 = \{m^3 : m^3 = (\mu^3, \sigma^3) \tag{24}$$

$$\sigma^3 \in \Re \tag{25}$$

$$\mu^3 : II_1 \times II_1 \rightarrow \Re \tag{26}$$

gdzie:

II_1 – zbiór indeksów: $II_1 = (-1, 0, +1)$.

$$\mu^3 = \{\mu^3(i,j)\} = \{\mu_{ij}^3\} = \begin{pmatrix} \mu_{-1,-1}^3 & \mu_{-1,0}^3 & \mu_{-1,1}^3 \\ \mu_{0,-1}^3 & \mu_{0,0}^3 & \mu_{0,1}^3 \\ \mu_{1,-1}^3 & \mu_{1,0}^3 & \mu_{1,1}^3 \end{pmatrix} \text{ gdzie } i,j \in II_1$$

W przedstawionym modelu m^3 ($m^3 \in M^3$):

- macierz μ^3 jest obrazem otoczenia agenta zawierającym wybrane informacje dotyczące sąsiednich węzłów, w szczególności ilości zasobu w tych węzłach,
- σ^3 reprezentuje ilość agentów typu $g = 1$ lub $g = 2$ wygenerowanych przez danego agenta typu $g = 3$.

Definicja konfiguracji strategii S^3: Konfiguracja strategii S^3 składa się ze strategii:

$$S^3 = \{s_a^3, s_b^3, s_{i,j}^3\} \text{dla } (i,j) \in \mathrm{II}_1 \times II_1 \setminus \{(0,0)\} \tag{27}$$

gdzie:

s_a^3 – strategia polegająca na wytworzeniu w węźle, w którym dany agent aktualnie się znajduje ($r_{k,\,l}$), agenta typu $g = 1$, związanego z tym węzłem, a następnie wykonanie operacji samolikwidacji danego wolnego agenta. Strategia ta ma następujący wpływ na zmianę modelu w „świadomości" agenta:

$$s_a^3(m) = s_a^3((\mu^3, \sigma^3)) = \tag{28}$$

$$= \begin{cases} (\mu^3, \sigma^{3'}) & \text{gdzie } \sigma^{3'} = \sigma^3 + 1 \quad \text{dla } \mu^3(0,0) < 0 \\ (\mu^3, \sigma^3) & \text{dla } \mu^3(0,0) \geq 0 \end{cases}$$

s_b^3 – strategia polegająca na wytworzeniu w węźle, w którym dany agent aktualnie się znajduje ($r_{k,\,l}$), agenta typu $g = 2$, związanego z tym węzłem, a następnie wykonanie operacji samolikwidacji danego wolnego agenta. Strategia ta ma następujący wpływ na zmianę modelu w „świadomości" agenta:

$$s_b^3(m) = s_b^3((\mu^3, \sigma^3)) = \tag{29}$$

$$= \begin{cases} (\mu^3, \sigma^{3'}) & \text{gdzie } \sigma^{3'} = \sigma^3 + 1 \quad \text{dla } \mu^3(0,0) > 0 \\ (\mu^3, \sigma^3) & \text{dla } \mu^3(0,0) \leq 0 \end{cases}$$

$s_{i,j}^3$ – strategia polegająca na przemieszczeniu się danego agenta znajdującego się aktualnie w węźle $r_{k,\,l}$ do odpowiednio wybranego węzła sąsiedniego $r_{k+i,\,l+j}$. Strategia ta ma następujący wpływ na zmianę modelu w „świadomości" agenta:

$$s_{i,j}^3(m) = s_{i,j}^3((\mu^3, \sigma^3)) = (\mu^{3'}, \sigma^3) \tag{30}$$

gdzie macierz opisująca otoczenie agenta μ^3 jest określona:

$$\mu^{3'}{}_{k,l} = \mu^{3'}(k, l) = \tag{31}$$

$$= \begin{cases} \mu^3(k+i, l+j) & \text{gdy } -1 \leq k+i \leq 1 i -1 \leq l+j \leq 1 \\ ? & \text{gdy zachodzi przeciwny przypadek} \end{cases}$$

gdzie: k, l: – $k \in II_1, l \in II_1$

i, j: – $(i, j) \in (II_1 \times II_1) \setminus \{(0, 0)\}$

$\sqrt{}$: – symbol nieokreślony.

Definicja konfiguracji celów Q^3 Konfiguracja celów (wskaźników jakości) Q^3 zawiera dwa elementy: q_a^3, q_b^3, z których każdy określa inny cel:

q_a^3 – odpowiada przypadkowi, w którym agent ma za cel wygenerowanie w węźle, w którym się aktualnie znajduje agenta typu $g = 1$ lub $g = 2$. Cel ten określa formuła:

$$q_a^3(m^3, m^3) = q_a^3((\mu^3, \sigma^3), (m^{3'}, \sigma^{3'})) = \tag{32}$$

$$= \begin{cases} \sigma^{3'} - \sigma^3 & \text{gdy } \mu^3(0,0) \neq 0 \\ 0 & \text{gdy } \mu^3(0,0) = 0 \end{cases}$$

g_b^3 – odpowiada przypadkowi, w którym agent ma za cel przemieszczenie się z węzła, w którym się aktualnie znajduje do jednego z węzłów sąsiednich, w którym znajduje się największa ilość zasobu (ze wszystkich węzłów sąsiednich). Określone jest to formułą:

$$q_a^3(m^3, m^{3'}) = q_a^3((\mu^3, \sigma^3), (m^{3'}, \sigma^{3'})) = \tag{33}$$

$$\begin{cases} 1 + \left| (\mu^{3'}(0,0) - \mu^3(0,0)) \right| & \text{gdy } \mu^3(0,0) = 0 \quad \left| (\mu^{3'}(0,0) - \mu^3(0,0)) \right| > 0 \\ 1 & \text{gdy } \mu^3(0,0) = 0 \quad \left| (\mu^{3'}(0,0) - \mu^3(0,0)) \right| = 0 \\ 0 & \text{gdy } \mu^3(0,0) \neq 0 \end{cases}$$

Powyższa formuła obejmuje przypadek, w którym agent ma za cel przemieszczenie się z węzła, w którym się znajduje do jakiegokolwiek węzła sąsiedniego. Ma to zastosowanie gdy zarówno węzeł, w którym agent się znajduje, jak i węzły sąsiednie nie spełniają warunków wymaganych przez danego wolnego agenta (np. nie posiadają zasobu) i agent musi kontynuować wędrówkę poprzez środowisko w jakimkolwiek kierunku.

Algorytm działania agenta typu $g = 3$ może zależeć od tego jaki jest porządek realizacji celów tzn. czy konfiguracja celów Q^3 jest uporządkowana i przeszukiwana cyklicznie. W opisywanym przypadku uporządkowanie celów nie ma znaczenia dla działalności agenta i różne uporządkowania nie powodują różnych algorytmów działania. Dla ustalenia uwagi można przyjąć następujące uporządkowanie celów:

- cel q_a^3,
- cel q_b^3.

Definicja funkcji (operatora) obserwacji I^3: Funkcja obserwacji I^3 określa w oparciu o zbiór modeli M^3 i rzeczywiste środowisko model tego środowiska w świadomości agenta. Stan środowiska, w którym aktualnie znajduje się agent określa stan węzła, w którym agent przebywa. Zatem jest to określone przez:

- macierz r reprezentującą środowisko grafowe,
- parę (k, l) określającą węzeł (wiersz i kolumnę), w którym znajduje się agent.

Zatem środowisko V można określić

$$V = (E, A, C) \tag{34}$$

gdzie:

E = $\{r_{ij}, r\ max_{ij}, r\ min_{ij}\}$ zbiór macierzy reprezentujący przestrzeń grafową (rozdz. 3),

A = $\varnothing$ zbiór agentów, z którymi dany agent (typu $g = 3$) może współdziałać – w rozważanym przypadku zbiór pusty, czyli agent nie współdziała z innymi agentami,

C = $\{(k, l)\}$ jednoelementowy zbiór zawierający parę (k, l), która określa węzeł (wiersz i kolumnę), w którym dany agent typu $g = 3$ aktualnie się znajduje i stanowi on dla niego „wirtualny" węzeł macierzysty.

Zatem funkcja I określa:

$$I^3(M^3, V = (\{r_{ij}, r\,max_{ij}, r\,min_{ij}\}, \varnothing, \{(k, l)\})) = m^3 = (\mu^3, \sigma^3) \tag{35}$$

gdzie:

$$s^3 = 0 \tag{36}$$

$$\mu(i, j) = \begin{cases} \rho(k+i, l+j) - r\,max(k+i, l+j) & \text{gdy } r(k+i, l+j) > r\,max(k+i, l+j) \\ \rho(k+i, l+j) - r\,min(k+i, l+j) & \text{gdy } r(k+i, l+j) < r\,min(k+i, l+j) \\ 0 & \text{w przeciwnym przypadku} \end{cases} \tag{37}$$

dla $i \in II_1, j \in II_1$.

Definicja funkcji (operatora) realizacji strategii X^3: Funkcja realizacji strategii X^3 jest dla zdefiniowanych strategii określona:

- $X^3(s_a^3, V = (\{r_{ij}, r\,max_{ij}, rmin_{ij}\}, \varnothing, \{(k, l)\}))$ – wygeneruj w węźle, w którym się znajdujesz (o współrzędnych (k, l)) agenta typu $g = 1$, związanego z tym węzłem, a następnie dokonaj samolikwidacji danego wolnego agenta,
- $X^3(s_b^3, V = (\{r_{ij}, r\,max_{ij}, r\,min_{ij}\}, \varnothing, \{(k, l)\}))$ – wygeneruj w węźle, w którym się znajdujesz (o współrzędnych *(k, l)*) agenta typu $g = 2$, związanego z tym węzłem, a następnie dokonaj samolikwidacji danego wolnego agenta,
- $X^3(s_{i,j}^3, V = (\{r_{ij}, r\,max_{ij}, r\,min_{ij}\}, \varnothing, \{(k, l)\}))$ – przemieść się do węzła reprezentowanego przez element macierzy r o numerze $k + i, l + j$ (kolumny i wiersza).

Przyjmując opisany algorytm działania, agent w oparciu o przedstawione definicje poszukuje węzła, w którym ilość zasobu jest większa od założonego maksimum (lub mniejsza od założonego minimum) i po znalezieniu generuje w tym węźle agenta typu $g = 1$ (lub $g = 2$), a następnie dokonuje operacji samolikwidacji.

4.2. Wyniki badań symulacyjnych systemu wieloagentowego z wolnymi agentami zastosowanego do rozpraszania zadań

Przedstawiona koncepcja zastosowania wolnego agenta została zbadana przy wykorzystaniu przykładowego systemu wieloagentowego rozpraszającego zadania w strukturach wieloprocesorowych. Wyniki badań symulacyjnych systemu przedstawiono w tabeli 1. W badanym systemie wykorzystuje się opisane w poprzednich rozdziałach 2, 3 trzy typy agentów a^0, a^1, a^2 oraz dodatkowo „wolnych" agentów typu $g = 3$. Badanie przedstawio-

nych wariantów systemów wieloagentowych przeprowadzone zostało drogą symulacji systemu rozpraszania zadań w strukturze wieloprocesorowej złożonej z 400 procesorów dla ustalonej grupy złożonej z 22 tysięcy zadań częściowych. Na początku obliczeń generowana była początkowa grupa złożona z około 10% ogólnej liczby zadań. Pozostałe zadania generowane były losowo jako efekt zakończenia liczenia już wygenerowanych zadań.

Tabela 1

Wyniki symulacyjnego badania wieloagentowego systemu rozpraszania zadań wariant systemu z agentami „wolnymi" typu $g = 3$ (wyniki 20 prób, x – wartość średnia, δ – odchylenie standardowe, d – przedział ufności, przyjęto poziom ufności 99 procent)

Nr	A	Nr	A
1	0,75163	11	0,75625
2	0,73602	12	0,7178
3	0,71808	13	0,71301
4	0,68143	14	0,71492
5	0,71969	15	0,71059
6	0,73644	16	0,71777
7	0,72490	17	0,7178
8	0,72197	18	0,69738
9	0,76048	19	0,73314
10	0,73752	20	0,74036
x	0,725359	σ	0,018719877
d	0,010778671	d%	1,48597747

Działający w systemie agenci wspomnianych typów, funkcjonują analogicznie jak w systemie bez agentów wolnych, ale dodatkowo poza wykonywanymi działaniami bezpośrednio związanymi z rozpraszaniem zadań, zajmują się pozyskiwaniem informacji o bieżącym stanie całego systemu w celu optymalizacji działania agentów i polepszeniem wydajności całego systemu.

Dodatkowo w badanym systemie agenci typów $g = 1$ i $g = 2$ w przypadku gdy nie mogą wykonać powierzonego im zadania, lub zakończą wykonywanie powierzonego zadania – nie ulegają likwidacji, ale przekształcają się w agenta „wolnego" typu $g = 3$.

W tabeli 2 przedstawiono porównanie różnych wariantów systemów, w szczególności:

- **wariant A** System wyłącznie z agentami rezydentnymi a^0 zapewniającymi przepływ zadań między sąsiednimi procesorami (węzłami) za pośrednictwem bezpośrednich połączeń między węzłami, bez transmisji za pośrednictwem magistrali,
- **wariant B1** System z agentami a^0 i agentami a^1 poszukującymi wolnych zadań dla bezczynnych procesorów. Przesyłanie zadań i agentów za pośrednictwem bezpośrednich połączeń między procesorami, a za pośrednictwem magistrali wyłącznie zadań. W systemie do ograniczania ilości agentów zastosowano mechanizm limitowania ilości generowanych agentów (rozdz. 3.1),

- **wariant B2** System z agentami a^0 i agentami a^2 poszukującymi bezczynnych procesorów w celu przesłania im wolnych zadań z procesorów posiadających ich nadmiar. Przesyłanie zadań i agentów za pośrednictwem bezpośrednich połączeń między procesorami, a za pośrednictwem magistrali wyłącznie zadań. W systemie do ograniczania ilości agentów zastosowano mechanizm limitowania ilości generowanych agentów (rozdz. 3.1),
- **wariant C** System z agentami a^0, agentami a^1 poszukującymi wolnych zadań dla bezczynnych procesorów i agentami a^2 poszukującymi bezczynnych procesorów w celu przesłania im wolnych zadań z procesorów posiadających ich nadmiar. Współpraca między agentami a^1 i a^2 w formie wymiany informacji. Przesyłanie zadań i agentów następuje za pośrednictwem bezpośrednich połączeń między procesorami, a za pośrednictwem magistrali wyłącznie zadań. W systemie do ograniczania ilości agentów zastosowano mechanizm limitowania ilości generowanych agentów przez agenta typu $g = 0$ (rozdz. 3.1),
- **wariant D** System z agentami a^0, agentami a^1 poszukującymi wolnych zadań dla bezczynnych procesorów, agentami a^2 poszukującymi bezczynnych procesorów w celu przesłania im wolnych zadań z procesorów posiadających ich nadmiar. Współpraca między agentami a^1 i a^2 w formie wymiany informacji. Przesyłanie zadań i agentów następuje za pośrednictwem bezpośrednich połączeń między procesorami, a za pośrednictwem magistrali wyłącznie zadań. W systemie do ograniczania ilości agentów zastosowano mechanizm w postaci koncepcji agentów „wolnych" typu $g = 3$ poszukujących pracy.

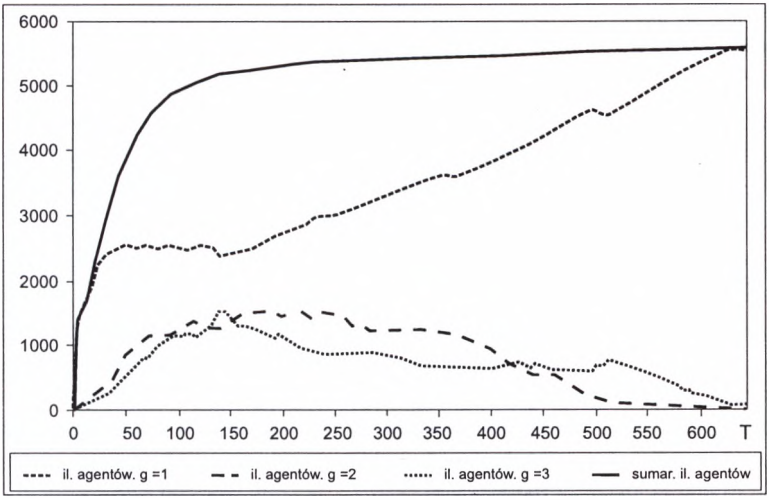
Tabela 2

Wyniki symulacyjnego badania wieloagentowego systemu rozpraszania zadań. Porównanie wydajności systemów dla różnych konfiguracji zastosowania agentów typów $g = 1$, $g = 2$ i $g = 3$. Oszacowanie estymatorów wykonane dla poziomu ufności 99 procent (x – wartość średnia, δ – odchylenie standardowe, d – przedział ufności)

	A	B2	B1	C	D
x	0,20013	0,58589	0,71531	0,73026	0,725359
σ	0,01626	0,02372	0,00785	0,00670	0,018720
d	0,00936	0,01366	0,00452	0,00386	0,010779
d%	4,67784	2,33127	0,63166	0,52807	1,485977

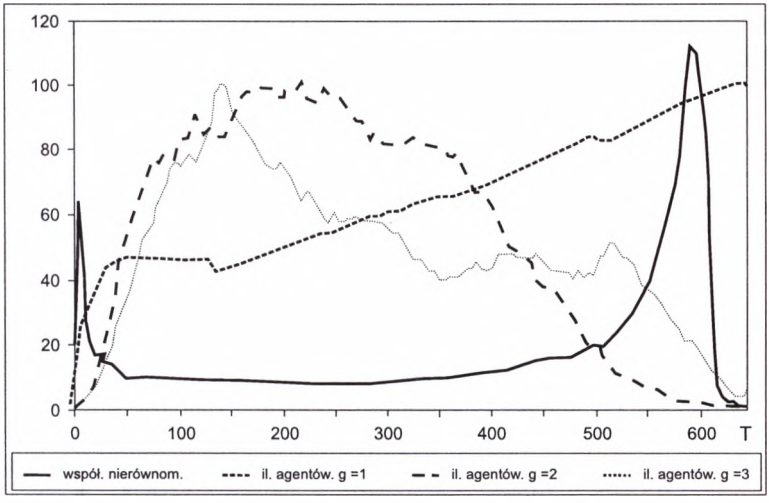
Wyniki przedstawione w tabeli 2 wykazują, że wydajność systemu z mechanizmem opartym na koncepcji agentów „wolnych" nie odbiega znacząco od wydajności z mechanizmem limitowania wygenerowanych agentów. Metoda wolnych agentów nie wymaga arbitralnego ustalania maksymalnych ilości agentów poszczególnych typów, a jedynie oszacowania maksymalnej ilości wszystkich agentów w systemie.

Wykres na rys. 9 przedstawia zmiany ilości agentów poszczególnych typów i sumaryczną ilość agentów w czasie działania systemu i wykonywania obliczeń zadań. Można zaobserwować, że sumaryczna ilość agentów narasta początkowo do ustalonej ilości i pozostaje praktycznie stała w czasie obliczeń. Natomiast ilości agentów poszczególnych typów zmieniają się w zależności od potrzeb. Fakt ten można również zaobserwować na wykresie (rys. 10), na którym porównano ilości agentów poszczególnych typów ze współczynnikiem nierównomierności rozłożenia zadań (wzór 2) w czasie obliczeń.



Rys. 9. Wykres przedstawiający ilość agentów w czasie (T). Wariant z agentami wolnymi („bezrobotnymi"). Na wykresie przedstawiono w pierwszej kolejności wykres współczynnika nierównomierności rozkładu zadań, oraz w następnej kolejności, względne ilości agentów typu $g = 1$, $g = 2$, $g = 3$ – agentów wolnych

Podsumowując można zauważyć, że zastosowanie mechanizmu opartego na koncepcji „wolnych" agentów daje wyniki porównywalne z pozostałymi przedstawianymi metodami ograniczania ilości agentów. Wydaje się jednak, że ogólniejsza koncepcja „wolnego" agenta może powodować, że mechanizm ten w praktycznych zastosowaniach będzie bardziej uniwersalny.



Rys. 10. Wykres przedstawiający ilość agentów w czasie (T). Wariant z agentami wolnymi („bezrobotnymi"). Na wykresie przedstawiono kolejno ilości agentów typu $g = 1$, typu $g = 2$, agentów typu $g = 3$ – agentów wolnych i wykres sumarycznej ilości agentów wszystkich typów ($g = 1, 2, 3$)

5. Zakończenie

W pracy przedstawiono przyczyny utraty integralności funkcjonalnej systemów wieloagentowych i propozycje rozwiązań, w szczególności ograniczenia nadmiernego wzrostu ilości agentów. Podsumowując można stwierdzić, że następujące zaproponowane rozwiązania zasługują na uwagę:

1. Problem nadmiernej ilości generowanych agentów nie może być bezpośrednio rozwiązany przez samych agentów odpowiedzialnych za tę generację, ponieważ nie dysponują oni informacją o charakterze globalnym, pozwalającą im na podjęcie odpowiednich decyzji.
   Istnieje jednak możliwość, że agenci mogą informację tę pośrednio uzyskać obserwując zachowanie pewnych grup agentów innych typów na tle całej społeczności agentów. Można nawet zasugerować rozwiązanie polegające na tym, że w systemie istnieją pewne typy agentów, których jedynym zadaniem jest dostarczanie pewnych informacji o charakterze globalnym innym agentom (także poprzez swoje specyficzne zachowanie się w systemie).
2. Stabilizacja liczby agentów przy wykorzystaniu koncepcji wolnych agentów wydaje się szczególnie obiecująca. Koncepcja ta może być rozważana jako specjalne podejście do systemu wieloagentowego, w którym to sami agenci poszukują dla siebie zadań (pracy), a nie dostają arbitralnie przydziału do „pracy". Ponadto grupa agentów wolnych zapewnia drogę do przepływu agentów z grupy danego typu do grupy innego typu, co powoduje samoczynną regulację liczności grup agentów poszczególnych typów w zależności od potrzeb systemu.
   Dodatkowo sterowanie zewnętrzne ilością agentów poprzez likwidację agentów wykonujących jakieś zadania w systemie powoduje, że powierzone im zadanie nie zostanie w całości wykonane – co może powodować dezorganizację systemu. Regulacja ogólnej liczby agentów powinna zatem odbywać się wyłącznie poprzez regulację liczby (likwidację, generację) agentów wolnych („bezrobotnych").

Literatura

[1] Cetnarowicz K.: *M-agent architecture based method of development of multiagent systems*. In *Proc. of the 8th Joint EPS–APS International Conference on Physics Computing*, ACC Cyfronet Krakow Poland, 1996

[2] Cetnarowicz K.: *Technology of decentralized multi–agent system creation based on the M–agent architecture*. In *Proc. MIFSYD'96, II–AGH, Leibniz/IMAG*, Technical Rep. No 11.1/96, Institute of Computer Science AGH Krakow, Poland, 1996

[3] Cetnarowicz E., Nawarecki E., Cetnarowicz K.: *Agent oriented technology of decentralized system based on the M-agent architecture*. In *Proc. of the MCPL'97, IFAC/IFIP conference*, CTI – Technological Center for Informatics Foudation, Campinas – SP, Brazil, LAG – Grenoble, France, BIBA, Bremen, Germany, 1997

[4] Cetnarowicz K., Nawarecki E., Żabinska M.: *M–agent architecture and its application to the agent oriented technology.* In *Proc. of the DAIMAS'97. International workshop: Disributed Artificial Intelligence and Multi–Agent Systems,* St. Petersburg, Russia, 1997

[5] Demazeau Y., Müller J.P., editors: *Decentralized A.I.* North–Holland ISBN 0–444–88705–9, 1990

[6] Demazeau Y., Müller J.P., Perram J., editors: *Modelling Autonomous Agent in a Multi-Agent World.* Odense University, 1994

[7] Wallach Y.: *Alternating Sequential/Parallel Processing.* Berlin, Springer–Verlag 1982

[8] Wooldridge M., Jennings N.: *Formalizing the cooperative problem solving proces.* In Demazeau et al. [7], 15–26

Recenzent
prof. dr hab. inż. Edward Nawarecki