Polska Akademia Nauk
Instytut Podstawowych Problemów Techniki

Wykorzystanie systemów wieloagentowych we współdziałaniu robotów mobilnych

mgr inż. Michał Gnatowski

Rozprawa doktorska

Promotor: Prof. dr hab. inż. Adam Borkowski

Warszawa, 2005

Spis treści

Rozdział 1

Wstęp

Robotyka mobilna jest stosunkowo młodą dziedziną intensywnie rozwijającą się od kilkunastu lat. Roboty mobilne znalazły zastosowanie między innymi w zadaniach: inspekcji, obróbki powierzchni (praca w warunkach niekorzystnych dla człowieka), eksploracji (miejsca trudno dostępne dla człowieka, np. przestrzeń kosmiczna), i tym podobne.
Bardzo efektownym przykładem robota mobilnego jest humanoidalny robot firmy Honda Motor Co. Prace rozpoczęte w 1986 roku zaowocowały powstaniem robotów o nazwie P2 i P3 w latach odpowiednio 1996 i 1997. Najnowszy produkt Hondy to robot „Asimo" (*ang. Advanced Step to Innovative Mobility*). Robot mierzy 120 cm i waży 43 kilogramy - to jest 40 cm i 87 kilogramów mniej niż jego poprzednik P3. Sztuczna inteligencja tego robota polega między innymi na rozpoznawaniu: poruszających się obiektów, gestów i twarzy, środowiska oraz dźwięków (rysunek 1.2).

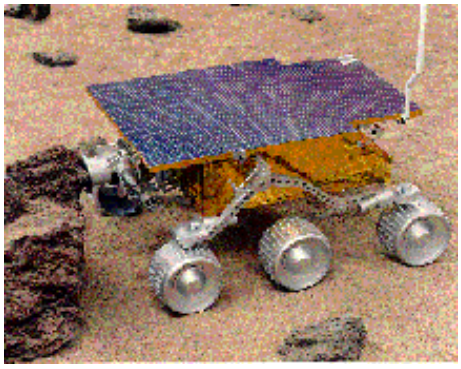
Rysunek 1.1: Humanoidalny robot *Asimo* firmy Honda Motor Co.

Mimo wielkich nakładów czasu i środków roboty Hondy nie znalazły jeszcze praktycznego zastosowania i cały czas powstają kolejne wersje robota. Przykładem rzeczywistego wykorzystania robotów są roboty *Rhino* i *Minerva* oprowadzające wyciekczki po muzeach. *Rhino* oprowadza po *Deutsches Museum Bonn*. Jest to robot powstały na Uniwersytecie

w Bonn w 1994 roku pod kierunkiem profesora Armina B. Cremersa. Ta sama grupa utworzyła *Minerva* - robota następnej generacji służącego do tego samego celu pracującego w *National Museum of American History* w Waszyngtonie [71].

Rysunek 1.2: Roboty *Rhino i Minerva* oprowadzające wycieczki po muzeach

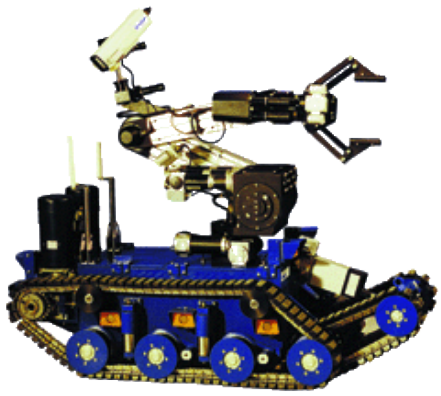
Bardzo dobrym przykładem wykorzystania robota w miejscu trudno dostępnym dla człowieka jest *The Mars Pathfinder Sojourner Rover* - robot eksplorujący powierzchnię Marsa podczas bezzałogowej wyprawy w latach 1996/97. Zbudowany przez zespół NASA pod kierunkiem Jacoba Matijevica i Donny Shirley, robot w trakcie wyprawy całkowicie autonomicznie przejechał około 100 metrów, zbadał około 250 metrów kwadratowych powierzchni planety i wykonał ponad 16 analiz chemicznych otaczających skał i ziemi. Ponadto w trakcie misji Pathfinder robił zdjęcia z kamery pokładowej, oraz mierzył temperaturę, ciśnienie i siłę wiatru [81]. Jest to pierwszy przykład autonomicznego działania w miejscu całkowicie niedostępnym ani dla człowieka, ani dla sygnału zdalnego sterowania (rysunek 1.3).



Rysunek 1.3: Robot *Pathfinder* eksplorujący powierzchnię Marsa

W Przemysłowym Instytucie Automatyki i Pomiarów została wykonana seria robotów interwencyjno-inspekcyjnych INSPECTOR. Ostatnia wersja robota SR-12 jest przedstawiona na rysunku 1.4. Jest to robot mobilny poruszający się na gąsienicach, wyposażony w szereg czujników. INSPECTOR jest urządzeniem zdalnie sterowanym za pomocą panela

operatorskiego. Jest wykorzystywany przez Policję i służy na przykład do zdalnego rozbrajania ładunków wybuchowych. Posiada możliwość autonomicznego działania, między innymi wycofanie się z miejsca akcji w przypadku braku sygnału zdalnego sterowania.



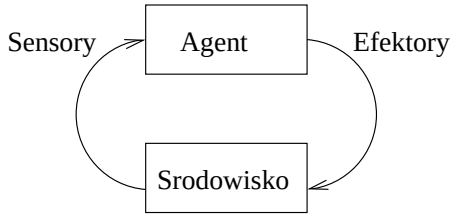
Rysunek 1.4: Robot *Inspector-12* zbudowany w PIAP

Powyższe przypadki oraz wiele innych przykładów pokazują, że autonomiczne roboty mobilne znajdują zastosowanie w praktyce i są pozytywnie przyjmowane przez społeczeństwo. Naturalnym rozwinięciem prac badawczych w robotyce, jest wykorzystanie zespołu robotów mobilnych w zadaniach niemożliwych do wykonania przez pojedynczego robota, lub wykonującym to zadanie mniej efektywnie. Sterowanie zespołem robotów jest zagadnieniem znacznie trudniejszym niż sterowanie pojedynczym robotem, ponieważ współpraca wielu robotów obejmuje dodatkowe dziedziny, takie jak: komunikacja pomiędzy robotami, koordynacja działań, negocjacje, itd.
W naukach informatycznych w ostatnich latach sporą popularność zdobyły systemy wieloagentowe. Są to systemy złożone z komunikujących się i współpracujących między sobą agentów, realizujących wspólne cele. Znajdują one zastosowanie w rozwiązywaniu problemów o charakterze rozproszonym lub złożonym obliczeniowo. Przykładami zastosowania systemów wieloagentowych są: wyszukiwanie informacji w sieci internet, symulowanie rynku, wspomaganie zarządzania w przedsiębiorstwach, zarządzanie sieciami telekomunikacyjnymi, lub kontrola ruchu lotniczego [105]. Traktując każdego robota jako jednego agenta, można w naturalny sposób wykorzystać mechanizm systemów wieloagentowych do rozwiązywania zadań związanych z zespołami wielu robotów. W niniejszej pracy podjęto próbę zastosowania systemów wieloagetnowych w rozwiązaniu zadania inspekcji znanego obszaru przez zespół wielu robotów.

1.1 Pojęcie agenta

Nie ma jednej precyzyjnej definicji agenta. Zwłaszcza różnica pomiędzy agentem a obiektem lub programem komputerowym jest nieprecyzyjna. W pracy opisującej architekturę

M-agenta, przestawionej np. w [22, 23] agent jest to element służący do budowy systemów zdecentralizowanych. Według tej koncepcji obiekt jest co prawda autonomiczny, ale nie ma inicjatywy działania, natomiast agent działa w oparciu o obserwacje otoczenia. Według innej definicji, przedstawionej np. w [113] opisującej system MRROC++, agentem jest cokolwiek, co postrzega środowisko i w jakiś sposób na nie oddziaływuje. Do dalszych rozwań można przyjąć, że agent jest systemem (programem) komputerowym usytuowanym w określonym środowisku, mogącym korzystać z określonych zasobów, zdolnym do autonomicznego działania, w celu osiągnięcia określonych celów i posiadającym motywację do działania. [108, 110, 37].
Powyższa definicja nie precyzuje co to jest środowisko, oraz co to jest autonomia agenta. Można przyjąć, że agent musi posiadać czujniki którymi jest w stanie odbierać sygnały wejściowe oraz efektory, którymi jest w stanie wpływać na otaczające go środowisko.



Rysunek 1.5: Agent w środowisku

Głównym i najtrudniejszym zadaniem agenta jest zdecydowanie który z możliwych ruchów w danym momencie agent powinien wykonać, w celu osiągnięcia określonego celu. Proces decyzyjny zależy oczywiście od warunków zewnętrznych czyli od środowiska, w których agent operuje. Środowisko pracy agenta może być następująco sklasyfikowane [96]:

- Dostępne / Niedostępne - określa dostępność informacji o środowisku. Im środowisko jest bardziej dostępne tym łatwiej zbudować agenta;
- Deterministyczne / Niedeterministyczne - deterministyczne środowisko oznacza, że każda akcja agenta ma jednoznacznie określoną, gwarantowaną odpowiedź środowiska;
- Epizodyczne / Nieepizodyczne - w środowisku epizodycznym zmiany środowiska nie zależą od działań agentów na innych scenach;
- Statyczne / Dynamiczne - w środowisku statycznym jest skończona liczba możliwych akcji i percepcji agenta. Przykładem środowiska statycznego może być gra w szachy;

Rzeczywisty świat jest: niedostępny, niedeterministyczny, nieepizodyczny i dynamiczny, co tłumaczy dlaczego tak trudno jest zbudować agenta operującego w realnym świecie.

Najprostrzym przykładem agenta może być termostat. Na podsatwie sygnału wejściowego (temperatura) jest generowane określone działanie (załączenie, lub wyłączenie ogrzewania). Zwykle jednak agent musi spełniać trzy cechy:

- reaktywność - agenci są w stanie zidentyfikować środowisko i zareagować w sposób umożliwiający realizację zadania;
- pro-aktywność - agenci są w stanie przejąć inicjatywę w celu realizacji zadania;
- zdolność współpracy - agenci są w stanie współdziałać ze sobą w celu realizacji zadania;

1.1.1 Obiekt i agent

W wielu przypadkach rola agenta w systemie wieloagentowym może być porównywana z obiektem w programowaniu obiektowym. Różnica polega na tym że [108]:

- agent posiada wewnętrzną świadomość i swój własny cel, który może być różny od celu innego agenta. Dlatego prośba od innego agenta zostanie wykonana tylko wtedy jeżeli jest ona zgodna z interesem pytanego agenta. Obiekty mają metody publiczne, które umożliwjają innym obiektom uruchamianie ich niezależnie od woli obiektu posiadającego taką metodę;
- agenci mają możliwość dostosowania swojego zachowania do sytuacji (reaktywność, pro-aktywność, zdolność współpracy). Obiekt też może mieć takie właściwości, ale są one niezmienne i nie zależą od stanu innych obiektów;
- każdy agent ma co najmniej jeden wewnętrzy wątek sterujący jego stanem wewnętrznym;

Poniższy przykład ilustruje różnicę pomiędzy obiektem i agentem (dla uproszczenia nie utworzono wątku sterującego agentem). Wykorzystano składnię języka *Java*.

OBIEKT

```
private int a,b;
public void setNewValues(int a, int b)
{
.  this.a = a; this.b = b;
}
```

AGENT

```
private int a,b;
public void setNewValues(int a, int b)
{
.  if (a>b) {
.  .  this.a = a; this.b = b;
.  .  sender.setConfirm(true);
.  else {
.  .  sender.setConfirm(false);
.  }
}
```

Ten bardzo prosty przykład pokazuje, że w obiekcie wartości się zmienia, natomiast agent może przyjąć lub odrzucić prośbę nadawcy. W tym przypadku, świadomość agenta polega na tym, że wartość „a" musi być większa od „b". Jeżeli nadawca (inny agent realizujący swój cel) prosi o zmianę wartości, prośba może być przyjęta lub odrzucona. W zależności od przyjętego modelu komunikacji, istnieje możliwość dalszej negocjacji i ewentualnych ustępstw. Rodzaje komunikacji będą omawiane w kolejnych rozdziałch. Inne rozważania na temat definicji agenta są przedstawione w: [96, 110, 76, 43, 55, 35, 42].

1.1.2 Agenci a systemy ekspertowe

W latach osiemdziesiątych systemy ekspertowe były najpopularniejszymi systemami metod sztucznej inteligencji [52]. Systemy ekspertowe (zwłaszcza działające w czasie rzeczywistym) są podobne do systemów wieloagentowych (na przykład *Archon* [54]). Główna różnica pomiędzy systemami ekspertowymi a systemami wieloagentowymi polega na pośrednim kontakcie systemu ekspertowego ze środowiskiem. Dane wejściowe nie pochodzą z sensorów, ale od dodatkowego pośrednika, którym przeważnie jest człowiek. Podobnie systemy ekspertowe nie oddziałują bezpośrednio na środowisko a jedynie udzielają rad lub w inny sposób wpływają na element pośredni pomiędzy systemem a środowiskiem.

1.2 Rodzaje agentów

Poniżej przedstawiono formalny opis agentów [108]. Załóżmy, że:
$S = \{s_1, s_2, ...\}$ - oznacza możliwe stany środowiska,
$A = \{a_1, a_2, ...\}$ - oznacza możliwe akcje agenta,
akcja może być traktowana jak funkcja:

$$action:\ S^* \rightarrow A \tag{1.1}$$

gdzie S^* oznacza wybraną sekwencję stanów środowiska S.

Środowisko może być zdefiniowane jako funkcja:

$$env:\ S \times A \rightarrow \rho(S) \tag{1.2}$$

co oznacza, że określony stan środowiska, pod wpływem akcji agenta jest odwzorowywany do nowego stanu.

Wpływ agenta na środowisko można przedstawić jako sekwencję zdarzeń (historię):

$$h = s_0 \rightarrow^{a(0)} s_1 \rightarrow^{a(1)} s_2 \rightarrow ... \rightarrow^{a(u-1)} s_u \rightarrow^{a(u)} ... \tag{1.3}$$

gdzie s_0 oznacza początkowy stan środowiska, a_u oznacza u. akcję wykonaną przez agenta, oraz s_u oznacza u. stan środowiska, który jest wynikiem akcji a_{u-1} w stanie s_{u-1}. Równanie 1.3 reprezentuje możliwą historię agenta, jeżeli spełnione są dwa warunki:

$$\forall\, u \,\epsilon\, \mathbf{N}, a_u = action\,(s_0, s_1, ..., s_u) \tag{1.4}$$

$$\forall\, u \,\epsilon\, \mathbf{N},\ \text{takich że}\ u > 0,\ s_u \,\epsilon\, env(s_{u-1}, a_{u-1}) \tag{1.5}$$

1.2.1 Agenci czysto reaktywni

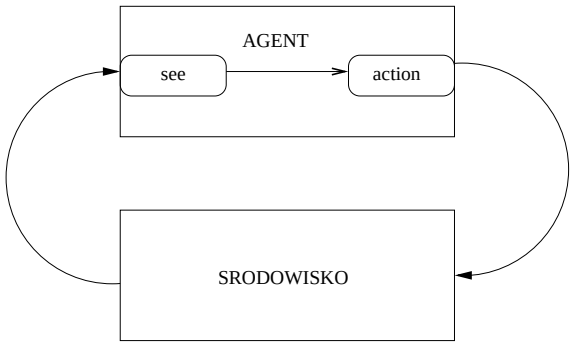
Pewna grupa agentów swoje decyzję opiera tylko na bieżącej wiedzy o otoczeniu, bez analizy poprzednich stanów środowiska. Taki typ agentów jest nazywany *czysto reaktywnymi agentami* (*ang. purely reactive*) [108]. Formalnie zachowanie agenta czysto reaktywnego może być opisane jako funkcja:

$$action:\ S \rightarrow A \tag{1.6}$$

Podawane przykłady agenta jako termostatu, lub przykład agenta z wartościami „a" i „b" są agentami czysto reaktywnymi. Funkcja *action* w termostacie można zapisać jako[108]:

$$action(s) = \begin{cases} \text{wyłącz, jeżeli } s \geq \text{ temperatura zadana} \\ \text{włącz w przeciwnym przypadku} \end{cases}$$

1.2.2 Percepcja agenta



Rysunek 1.6: Agent w środowisku

Poza funkcją *action*, agent musi mieć zdefiniowaną funkcję do odczytywania sygnałów wejściowych. Do tego służy funkcja *see*:

$$see:\ S \rightarrow P \tag{1.7}$$

która odwzorowuje stany środowiska w percepcje P. W związku z tym, funckja *action* przyjmuje postać:

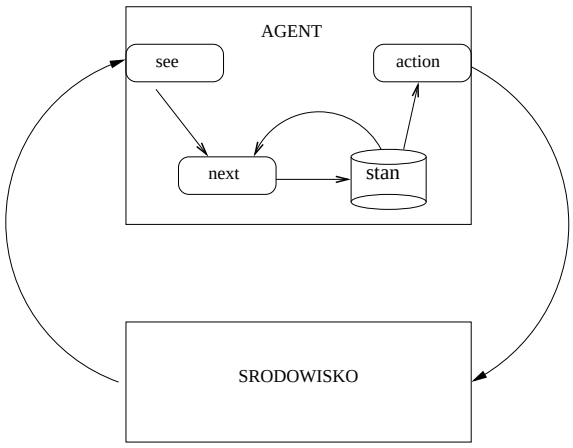
$$action:\ P^* \rightarrow S \tag{1.8}$$

która odwzorowuje sekwencje percepcji do akcji. Tak zdefiniowany schemat agenta wygląda tak jak przedstawiono na rys.1.6

1.2.3 Agent z parametrem wewnętrznym

W tym podrozdziale będzie przedstawiony model agenta, który do swojej funkcji decyzyjnej *action* wykorzystuje poprzednie stany środowiska (historię), co zostało opisane równaniem 1.3.

Schemat agenta z parametrem wygląda tak jak przedstawiono na rys.1.7 [108]



Rysunek 1.7: Agent z parametrem wewnętrznym

Funkcja percepcyjna *see* opisana równaniem 1.7 pozostaje niezmieniona. Funkcja decyzyjna *action* zostaje opisana równaniem:

$$action:\ I \rightarrow A \tag{1.9}$$

gdzie I oznacza stan wewnętrzny.
Dodatkowo zostaje zdefiniowana funkcja *next*, która odwzorowuje parametr wewnętrzny i percepcję w parametr wewnętrzny:

$$next:\ I \times P \rightarrow I \tag{1.10}$$

Zachowanie takiego agenta może być opisane następująco: Agent rozpoczyna z pewnym stanem wejściowym i_0. Otrzymuje dane wejściowe s i tworzy funkcję *see(s)*. Następnie funkcja *next* uaktualnia parametr wewnętrzny $next(i_0, see(s))$. Funkcja decyzyjna generowana przez agenta jest opisana funkcją $action(next(i_0, see(s)))$. Przy kolejnych odczytach danych wejściowych proces jest powtarzany.

1.3 Architektury systemów wieloagentowych

W poprzedniej części rozdziału agenta traktowano jako pojęcie abstrakcyjne. Zamodelowano funkcję decyzyjną agenta, nazwaną: *action*, która potrafi zdecydować, którą z możliwych czynności efektorów mają wykonać agenci. W niniejszym rozdziale zostanie

omówione, jak funkcja *action* może być zaimplementowana.
Rozróżniamy cztery rodzaje agentów [108]:

- agent logiczny - funkcja decyzyjna jest implementowana w wyniku dedukcji;
- agent reaktywny - funkcja decyzyjna jest implementowana na podstawie analizy określonego stanu środowiska (mapowanie funcji *situaction* do *action*).
- agent bdi (*przekonanie-chęć-zamiart*) - funkcja decyzyjna zależy od kombinacji danych reprezentowanych jako: *beliefs, desires, intentions.*
- architektury wartswowe - funkcja decyzyjna jest realizowana przez softwarowe warstwy, w których każda odpowiada za inny poziom abstrakcji środowiska.

1.3.1 Architektury oparte na logice

W niniejszym podrozdziale przedstawiono tradycyjne podejście od zagadnień sztucznej inteligencji, w którym dane wejściowe agenta są przedstawiane w postaci symoblicznej, reprezentowanej jako formuły logiczne a proces decyzyjny agenta jest logiczną dedukcją. Jako przykład rozważmy agenta logicznego, w którym stan wewnętrzny jest bazą danych zawierającą formuły opisane w logice pierwszego rzędu, np. [108]:
. *Open (valve 221)*
. *Temperature (reactor1234, 321)*
. *Pressure(tank776, 28)*
W tym przypadku baza danych jest informacją agenta o stanie otoczenia. Nie musi to być stan rzeczywisty środowiska. Jeżeli w bazie danych jest zapis że np. zawór nr 221 jest otwarty, to oznacza że agent wierzy, że ten zawór jest otwarty. W rzeczywistości albo może to być prawda, albo z różnych względów agent mógł otrzymać złą informację (np. uszkodzony czujnik).

Parametr wewnętrzny można traktować jako bazę wiedzy na temat wierzeń o otoczeniu, funkcję *see, next i action* można zapisać następująco (patrz rys.1.7):

$$see:\ S \rightarrow P \tag{1.11}$$

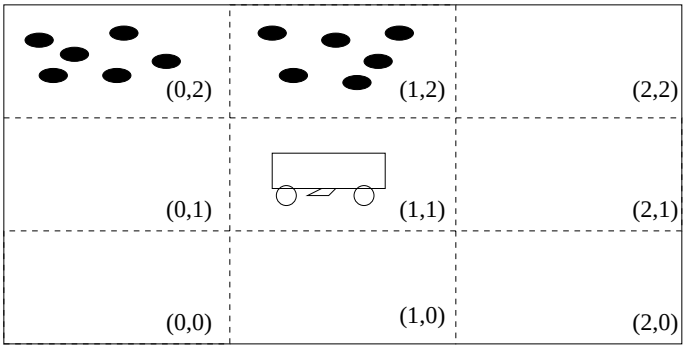
$$next:\ D \times P \rightarrow D \tag{1.12}$$

$$action:\ D \rightarrow A \tag{1.13}$$

gdzie S - sygnały wejściowe agenta, P - percepcja, D - baza danych, A - akcja agenta. W tym przypadku po odczytaniu danych wejściowych i odwzorowaniu w percepcji, następuje aktualizacja bazy danch o środowisku *beliefs*, poprzez odwzorowanie percepcji P i aktualnej bazy danych D do nowej bazy danych D (funkcja *next*, równanie: 1.12). Po aktualizacji bazy danych agent generuje funcję decyzyjną *action*, odwzorowując bazę danych D

w określoną akcję (funkcja *action*).

Przykład [108, 96]:
Mamy robota, którego zadaniem jest odkurzanie zadanej powierzchni (sceny). Robot wyposażony jest w sensor, który daje informację czy jest kurz, czy go nie ma. Orientacja robota jest zawsze zdefiniowana i jest jedną z wartości: *north, south, east, west.* Robot może w jednym kroku poruszać się o jedną klatkę do przodu, lub obrócić siç o 90°. Scena jest podzielona na ponumerowane klatki. Zakładamy, że agent (robot) jeździ po scenie i jedynym jego zadaniem jest odkurzanie. Robot nigdy nie opuszcza sceny. Dla uproszczenia zakładamy, że scena została podzielona na 3 x 3 klatki. Schemat zadania przedstawia rysunek 1.8 . Zakładamy, że robot zaczyna od klatki (0,0), z orientacją *north.*



Rysunek 1.8: Przykład struktury robota czyszczącego powierzchnię

Sygnałem wejściowym dla robota jest informacja czy w danej klatce jest kurz czy nie (*dirt*, lub *null*). Sygnałem wyjściowym (czyli funkcją *action*) jest jedna z możliwości: *forward, suck, turn*, czyli jazda do przodu, czyszczenie lub obrót. Zadanie polega na optymalnym dobraniu algorytmu pracy robota.
Rozwiązanie zadania:
1. Tworzymy 3. predykaty:
. *In(x,y)* - robot jest w klatce (x,y);
. *Dirt(x,y)* - w klatce (x,y) jest kurz;
. *Facing (d)* - robot jest zwrócony w stronę d;
2. Definiujemy funkcję *next*:
Funkcja musi odczytać dane wejściowe (*dirt* lub *null*) i zapisać nowe dane do bazy danych, oraz usunąć stare lub nieprawdziwe informacje. Ponadto funkcja musi wyliczyć nową lokalizcję i orientację agenta. Funkcję *next* wyznaczamy w kilku krokach.
Krok 1: wyznaczenie funkcji *old(Δ)* określającą stare dane w bazie danch, które chcemy uaktualnić. Δ oznacza element bazy danch D, P - percepcja agenta.

$$old(\Delta) = \{P(t_1, ..., t_n) | P \epsilon \{In, Dirt, Facing) \text{ oraz } P(t_1, .., t_n) \epsilon \Delta\} \qquad (1.14)$$

Krok 2. Tworzymy funkcję *new*, która dodaje zbiór nowych predykatów do bazy danych.

$$next:\ D \times P \rightarrow D \tag{1.15}$$

Krok 3. Na podstawie kroku 1 i 2, tworzymy funkcję *next*:

$$next(\Delta, p) = (\Delta \backslash old(\Delta)) \cup new(\Delta, p) \tag{1.16}$$

3. Określamy reguły rządzące zachowaniem agenta. Najważniejszą regułą z punktu widzenia zadania robota jest następująca reguła:

$$In(x, y) \wedge Dirt(x, y) \rightarrow Do(suck) \tag{1.17}$$

co oznacza, że jeżeli robot rozpozna kurz w klatce, w której się znajduje, to ma odkurzać. W innym wypadku robot ma jechać, lub obrócić się. Załóżmy, że robot ma poruszać się w następujący sposób: zaczyna od klatki (0,0) do (0,1), do (0,2), następnie do (1,2), (1,1) itd. Po dojechaniu do klatki (2,2) robot wraca do klatki (0,0). Rządzą tym następujące reguły:

$$In(0,0) \wedge Facing(north) \wedge \neg Dirt(0,0) \rightarrow Do(foreward) \tag{1.18}$$

$$In(0,1) \wedge Facing(north) \wedge \neg Dirt(0,1) \rightarrow Do(foreward) \tag{1.19}$$

$$In(0,2) \wedge Facing(north) \wedge \neg Dirt(0,2) \rightarrow Do(turn) \tag{1.20}$$

$$In(0,2) \wedge Facing(east) \rightarrow Do(foreward) \tag{1.21}$$

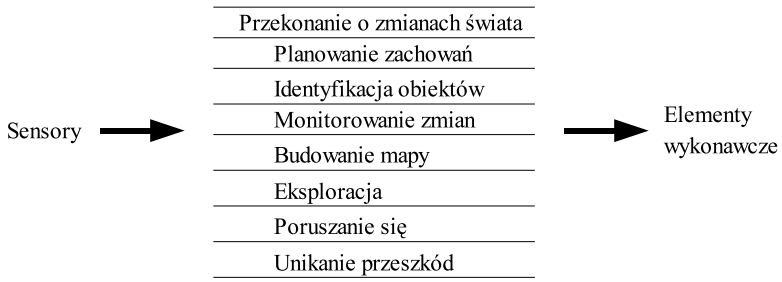
Każda reguła musi sprawdzić, czy regula 1.17 zachodzi czy nie. Do pełnego opisu agenta potrzeba zdefiniować reguły opisujące zachowanie agenta w pozostałych klatkach oraz mechanizm przejścia z klatki (2,2), do klatki (0,0). łatwo pokazać, że pełen zestaw reguł razem z funkcją *next* gwaratnuje żądane zachowanie agenta (robota).
Powyższy sposób realizowania agentów jest elegancki i ma przejrzystą semantykę. Wadą jednak jest złożoność obliczeniowa a co za tym idzie długi czas wyboru optymalnego działania, co w problemach czasu rzeczywistego jest nie do zaakceptowania [44]. Dlatego inne sposoby realizacji agentów są omówione w kolejnych podrozdziałach.
Dalsze rozważania na temat architektur opartych na logice można znaleźć w: [44, 62, 109]. Opis systemu CONGOLOG opisujący podejście stosunkowo „czysto" logiczne jest przedstawiony w [65]. Język programowania METATEM i Concurrent METATEM zaproponowany przez Fishera i inn. przedstawiono w [13, 41].

1.3.2 Architektury reaktywne

Symboliczna reprezentacja otoczenia nie sprawdziła się z powodu silnych ograniczeń czasowych, które występują w rzeczywistym świecie. W związku z tym w drugiem połowie lat 80. rozpoczęto poszukiwania alternatywnych sposobów do symbolicznej reprezentacji otoczenia. Trzy gówne postulaty tego podejścia to [108]:

- rezygnacja z symbolicznej reprezentacji otoczenia, oraz generowania funkcji decyzyjnej *action* na podstawie tej reprezencacji;
- racionalne zachowanie jest sprzężone ze środowiskiem, w którym działa agent i jest wynikiem interakcji pomiędzy agentem i środowiskiem;
- inteligentne zachowanie składa się z sumy innych prostych inteligentnych zachowań;

Przykładem takiej architektury może być architektura Rodney'a Brooks'a - *Subsumption* [19, 108, 106] (rys.1.9). Funkcja decyzyjna jest realizowana przez określony zbiór



Rysunek 1.9: Schemat architektury typu Subsumption

zachowań, z których każdy cały czas czyta sygnał wejściowy i generuje własną funkcję *action*. W propozycji Brooks'a każde zachowanie jest reprezentowane przez automat skończony. Ważnym elementem tej architektury jest to, że nie występuje symboliczna reprezentacja otoczenia. Każde zachowanie jest reprezentowane przez oddzielną warstwę. Ponieważ każda warstwa generuje swoją funkcję *action* niezależnie od innych warstw, musi istnieć mechanizm wskazujący, która funkcja *action* w danym kroku będzie wykonana. W architekturze typu *Subsumption* niższe warstwy mają wyższy priorytet, wskutek tego blokują wyższe warstwy. Im wyższa warstwa tym reprezentuje bardziej abstrakcyjny poziom zachowań. Rozważmy następujący przykład [103, 108]:
Agent jest pojazdem do zbierania określonego typu skał na obcej planecie. Lokalizacja tych skał nie jest znana, ale wiadomo, że występują grupami. Pewna grupa agentów porusza się po scenie w poszukiwaniu skał a znalezione skały zwozi do pojazdu - matki. Mapa otoczenia jest nieznana, ale wiadomo, że na scenie jest wiele przeszkód, wzgórz itp, które w istotny sposób utrudniają komunikację pomiędzy agentami a pojazdem - matką.
Zadanie polega na zaproponowaniu architektury dla systemu w celu efektywnego zbierania skał. Steels wprowadził dwa mechanizmy. Po pierwsze, w celu orientacji agenta względem pojazdu-matki, pojazd-matka wysyła sygnał radiowy, którego siła maleje wraz z odległością (*ang. gradient field*). Sygnał nie zawiera żadnych informacji. Jeżeli agent chce jechać do pojazdu-matki, wystarczy, że będzie jechał w stronę silniejszego sygnału. Drugi mechanizm dotyczy komunikacji pomiędzy agentami. Ukształtowanie terenu zgodnie z założeniami utrudnia bezpośrednią komunikację radiową, w związku z tym zaproponowany mechanizm polega na komunikacji pośredniej. Każdy agent posiada pewną

ilość radioaktywnych „okruchów", które może zostawiać, wykrywać lub zbierać ze sceny. Jeżeli agent zostawi gdzieś takie okruchy, później inny agent będzie mógł je zebrać. W celu ustalenia algorytmu działania całego systemu, najpierw określamy algorytm dla pojedynczego agenta, a następnie rozbudujemy algorytm dla całego zespołu.
Dla pojedynczego robota (bez żadnej współpracy) najniższa warstwa (a więc o najwyższym priorytecie) jest odpowiedzialna za omijanie przeszkód. Zachowanie to można opisać regułą:

Jeżeli wykryłem przeszkodę z przodu, to zmieniam kierunek. (1.22)

Druga warstwa jest odpowiedzialna za zbieranie skał i zwożenie ich do statku-matki. Jej reguły mogą wyglądać następująco:

Jeżeli mam obiekt i jestem w bazie to zostawiam obiekt. (1.23)

Jeżeli niosę obiekt i nie jestem w bazie to idę w stronę silniejszego sygnału. (1.24)

Trzecia warstwa odpowiada za zbieranie skał, które agent znajdzie:

Jeżeli wykryłem skałę, to ją podnoszę. (1.25)

Czwarta warstwa uruchamia się, gdy żaden poprzedni warunek nie został spełniony i powoduje że agent porusza się ruchem przypadkowym:

Jeżeli „prawda" to ruch losowy. (1.26)

Jak już wspomniano, każda warstwa ma określony priorytet, niższe warstwy blokują wyższe, co przedstawia poniższa zależność:
(1.22) < (1.23) < (1.24) < (1.25)< (1.26)
Powyższy przykład pokazuje, że tak skonstruowany agent będzie pracował prawidłowo. Agent zawsze ominie przeszkodę, po znalezieniu skały podniesie ją, pójdzie w stronę statku-matki i odłoży skałę. Z założeń zadania wiemy, że szukane skały występują grupami. W związku z tym, przydatny byłby mechanizm komunikacji. Jeżeli jeden agent znajdzie miejsce, gdzie znajduje się grupa szukanych skał, powinien poinformować o tym pozostałych agentów. Z założeń wiadomo również, że bezpośrednia komunikacja jest niemożliwa. Dlatego Steels zaproponował mechanizm oparty na algorytmach mrówkowych. Po znalezieniu skały agent wracając do bazy zostawia za sobą ślad w postaci radioaktywnych okruchów. Jeżeli jakiś czas później inny agent spotka taki ślad, wystarczy, że będzie jechał w stronę malejącego sygnału, żeby trafić do grupy szukanych skał. Poruszając się w stronę słabszego sygnału agent zbiera okruchy, osłabiając w ten sposób ślad. Natomiast idąc w stronę statku-matki, agent zostawia okruchy, powiększając ślad. Jeżeli zasoby szukanych skał wyczerpią się, agenci idąc w poszukiwaniu skał stosunkowo szybko usuną już nieaktualną ścieżkę. Reguły opisujące powyższe zachowanie są następujące:

Jeżeli niosę obiekt i jestem w bazie to zostawiam obiekt. (1.27)

Jeżeli niosę obiekt i nie jestem w bazie to odkładam 2 okruchy i idę w stronę silniejszego sygnału. (1.28)

Reguła opisująca jazdę do szukanych skał z podnoszeniem okruchów:

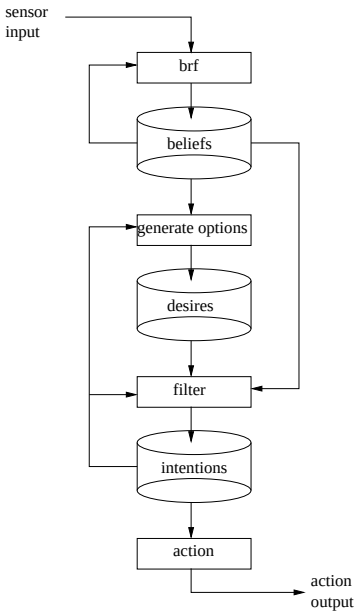
Jeżeli wykryłem okruch, podnoszę jeden okruch i jadę w stronę malejącego sygnału (1.29)

W tym przypadku hierarchia zależności pomiędzy warstwami jest następująca:
(1.22) < (1.27) < (1.28) < (1.25)< (1.29) < (1.26)
Steels twierdzi, że takie zachowanie grupy agentów jest w miarę optymalne, tanie (każdy agent wymaga minimalnej ilości mocy obliczeniowej) i krzepkie - strata jednego agenta nie wpływa na zachowanie reszty grupy. Poza wieloma zaletami, istnieją pewne nierozwiązane dotychczas zagadnienia związane z architekturą typu *subsumption*. Agent nie posiada modelu środowiska, przez co musi mieć wystarczającą ilość informacji zawartą w swoim lokalnym środowisku. W związku z tym trudno jest określić jak lokalna decyzja agenta wpływa na globalne zachowanie grupy. Trudno jest określić jak taki reaktywny agent mógłby sam się uczyć. Nie istnieje metodologia do budowania agentów, ponieważ nie są znane relacje pomiędzy zachowaniem agenta, zachowaniem grupy i środowiskiem. W praktyce okazuje się, że dziesięć warstw najczęściej jest maksymalną ilością do efektywnej pracy agenta, a budowanie agentów z większą ilością warstw okazuje się zbyt skomplikowane do praktycznego wykorzystania [36].
Najpopularniejszą architekturą reaktywną jest architektura Brooks'a opisana powyżej. Przegląd architektur reaktywnych jest przedstawiony w [67, 2]. Inne publikacje na ten temat to: sieciowa architektura agenta (*ang. agent network architecture*) zaproponowana przez Maes'a [69, 68, 70]; *teleo reactive programs* (Nilsson)[78]; wykorzystanie mechanizmu automatów skończonych (Rosenchein i Kaelbling) [94, 56, 57, 95]; system PENGI (Agre i Chapman)[1]; użycie drzew decyzyjnych, pozwalających na podjęcie odpowiedniej akcji (Schopper) [97]; pakiety akcji reaktywnych (Firby)[39].
Jedną z nowszych propozycji jest architektura *L*OGUE zaproponowana przez Takahashiego i innych [107]. Polega ona na podziale informacji robota na obiekty behavioralne (*ang. Behavior Element Object (BEO)*) i obiekty zadaniowe (*Task Object (TO)*), w sensie programowania obiektowego (OOP). Wykorzystując mechanizm JavaRMI obiekty te mogą są wymieniane pomiędzy robotami i serwerem behavioralnym.

1.3.3 Architektury typu BDI

Architektury typu BDI *belief-desire-intention*, (przekonanie-pragnienie-intencja) mają swoje filozoficzne korzenie w procesie określanym jako *practical reasoning*, zaproponowanym przez Michaela Bratmana [17]. Polegają one na określaniu krok po kroku jaką decyzję w danym momencie należy podjąć w celu osiągnięcia określonego celu. Proces ten posiada dwa główne elementy: pierwszy określa **jaki cel** chcemy osiągnąć (*deliberation*), drugi określa **jak** zamierzamy to zrobić (*means-ends reasoning*). Kluczową rolę w architekturze typu *BDI* odgrywają intencje. Kierują one procesem *means-ends*, (jeżeli in-

tencją agenta jest dojechanie do określonego miejsca, agent zastanawia się jak może to osiągnąć). Intencje ograniczają proces decyzyjny (jeżeli agent chce dojechać do określonego punktu, to nie zastanawia się jak zrealizować inne cele). Intencje „upierają się" (jeżeli agent chce dojechać od określonego punktu a nie jest to możliwe w jeden sposób, agent szuka innego sposobu. Bez ważnej przyczyny nie „poddaje się"). Intencje wpływają na przyszłe wierzenia (*beliefs*) (jeżeli agent chce dojechać do określonego punktu, to może zakładać, że już tam jest i rozpatrywać kolejne kroki). Dobre zaprojektowanie architektury typu *BDI* polega na znalezieniu kompromisu pomiędzy powyższymi wykluczającymi się ograniczeniami. Jeżeli np. agent stwierdzi, że nigdy nie osiągnie określonego celu (np. nie dojedzie do określonego punktu), lub osiągnięcie określonego celu przestaje być istotne, agent powinien zweryfikować swoje intencje. Z drugiej strony zbyt częsta zmiana intencji, powoduje że agent wykonuje swoje działania bezproduktywnie. Tego typu



Rysunek 1.10: Schemat architektury typu BDI

badania zostały przeprowadzone przez Davida Kinny i Michaela Gergeffa w implementacji architektury *BDI* - **dMARS**. Podzielili oni agentów na dwa rodzaje: *bold* i *cautions*. Agenci typu *bold* nigdy nie weryfikują swoich intencji, agenci typu *cautions* bardzo często zatrzymują swoje działanie w celu weryfikacji intencji. Kinny i Gergeff wprowadzili nowy parametr: γ, określający opis zmiany świata (*rate of world change*). Jeżeli γ jest duże, czyli środowisko zmienia się często, bardziej efektywni są agenci typu *cautions*, ponieważ są w stanie dopasować się do środowiska. Jeżeli γ jest małe, czyli środowisko nie zmienia się szybko, agenci typu *bold*, są efektywniejsi, gdyż nie tracą czasu na weryfikację swoich intencji. Opis formalny architektury **dMARS** znajduje się w [30].
Architektura typu *BDI* składa się z siedmiu elementów, co przedstawia rysunek: 1.10 [108]:

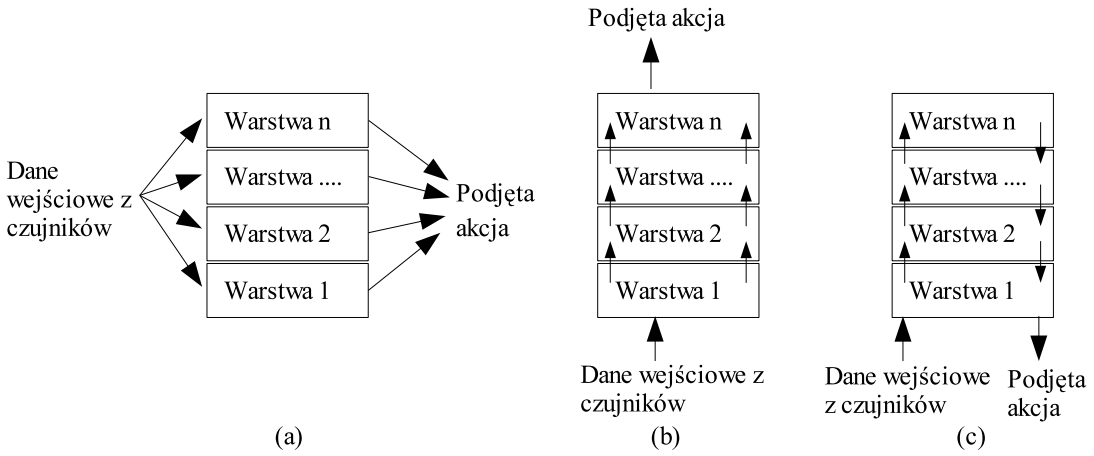
- funkcja *brf* (*belief revision function*), która na podstawie danych wejściowych modyfikuje stare i wprowadza nowe wierzenia (*beliefs*);
- zbiór wierzeń o otaczającym świecie (*beliefs*);
- funkcja generowania opcji (*option generation function*), określa opcje dostępne dla agenta (*desires*) na podstawie aktualnych wierzeń (*beliefs*) i intencji (*intentions*);
- zbiór dostępnych opcji (*desires*);
- funkcja filtrująca, która reprezentuje proces decyzyjny agenta i określająca jego intencje;
- zbiór intencji agenta (*intentions*);
- funkcja wyboru akcji (*action selection function*);

Architektury typu *BDI* są atrakcyjne z wielu powodów. Podstawowym elementem jest to, że są intuicyjne. Decydowanie co chcemy robić, (czyli jaki mamy cel), oraz określanie jak chcemy to osiągnąć, jest bliskie naturze człowieka. Ponadto architektury typu *BDI* jednoznacznie pokazują, jakie części systemu będą potrzebne do zbudowania agenta. Zadanie polega na odpowiednim zaimplementowaniu tych funkcji.
IRMA - jedną z architektur BDI jest przedstawiona w [18]. M.Georgeff i A.Lansky zaproponowali system wnioskowania proceduralnego (*ang. Procedural Reasoning System - PRS*) [46], który został użyty w praktyce w systemie kontroli ruchu lotniczego w Sydney - system OASIS[45].
Architektury BDI zostały w dużym stopniu sformalizowane między innymi przez: Cohena i Lavesquea [27], Singha [101], oraz Rao i Georgeffa [84, 86, 87, 88, 85]. Model komunikacji pomiędzy agentami został opisany w [51]. W Polsce badania nad architekturą *BDI* prowadzone są między innymi przez Barbarę Dunin-Kęplicz w Instytucie Informatyki Uniwersytetu Warszawskiego [32].

1.3.4 Architektury warstwowe

Każda architektura posiadająca co najmniej dwie warstwy jest architekturą warstwową. Warstwy mogą być ułożone względem siebie równolegle (*horizontal layering*) lub hierarchicznie (szeregowo) (*vertical layering*). W architekturach równoległych każda wartstwa odbiera sygnały wejściowe, przetwarza je i generuje sygnał wyjściowy. Zaletą tego typu architektur jest szybkość działania, wadą jest konieczność istnienia mechanizmu decyzyjnego, który będzie decydował, którą wartstwę w danym kroku aktywować. Przykład architektury równoległej jest opisany np. w [106]. W architekturach hierarchicznych, dane wejściowe są odbierane tylko przez jedną warstwę a następnie (w zależności od typu architektury i konkretnej sytuacji) dane są przesyłane do kolejnych warstw. Zaletą tego typu architektur jest brak mechanizmu decyzyjnego i prostota działania, wadą jest wolniejsze działanie od architektur równoległych, spowodowane dłuższą drogą przepływu

sygnaów. Bardzo popularną architekturą hierarchiczną jest architektura *InteRRaP* która jest omówiona w kolejnym podrozdziale. Przykładem architektury równoległej jest zaproponowana przez Brooks'a architektura *Subsumption* omówiona w punkcie 1.3.2. Schemat architektury równoległej i hierarchicznej przedstawia rysunek 1.11 [72].

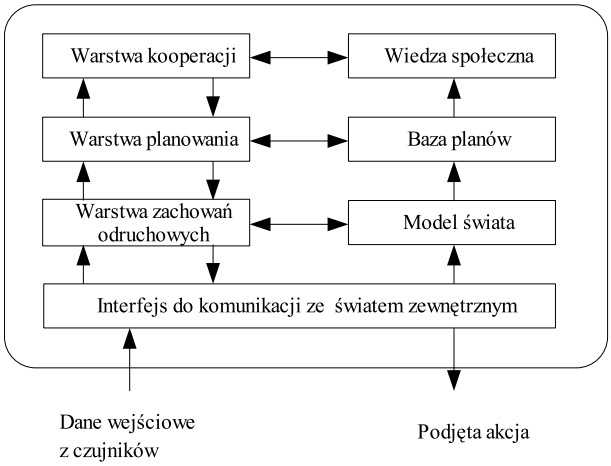


Rysunek 1.11: Przepływ sterowania w: a) architekturze równoległej; b) architekturze szeregowej, z przepływem jednokierunkowym; c)architekturze szeregowej, z przepływem dwukierunkowym.

InterRRaP

Architektura typu InteRRaP jest przykładem szeregowej architektury z przepływem sterowania w dwióch kierunkach. Najniższa warstwa *(behaviour layer)* odpowiada za zachowania odruchowe, środkowa warstwa *(plan layer)* odpowiada za planowanie, natomiast najwyższa warstwa *(cooperative layer)* odpowiada za współpracę z innymi agentami. Każda warstwa ma odpowiadającą jej bazę danch, która reprezentuje stan środowiska odpowiadający danej warstwie. Schemat architektury typu InteRRaP przedstawia rysunek 1.12.

Interakcja pomiędzy warstwami jest dwukierunkowa. Najpierw sygnał sterujący jest przesyłany od warstw niższych do wyższych (*bottom-up activation*), później od warstw wyższych do niższych (*top-down execution*). Jeżeli niższa warstwa korzystając ze swojej bazy danch nie jest w stanie podjąć odpowiedniego działania, przysyła sygnał do warstwy wyższej. Stąd wniosek, że im wyższa warstwa tym rzadziej jest aktywowana. Można to porównać do człwoieka idącego ulicą. W każdym kroku człowiek uważa żeby się nie przewrócić i z niczym nie zderzyć (odpowiednik najniższej warstwy - reaktywnej), natomiast gdzie ma iść analizuje tylko będąc na rozdrożu (odpowiednik warstwy planowania). Odpowiednikiem warstwy najwyższej może być decyzja żeby w ogóle wyjść z domu, w określonym celu. Wykorzystanie architektury InteRRaP jest omówione np. w [75, 40, 74].



Rysunek 1.12: Przepływ sterowania w architekturze typu InteRRaP.

Zastosowanie architektur wielowarstwowych

Architektura szeregowa do sterowania ruchem maszyny kroczącej została przedstawiona w pracy [112]. W pracach [49, 63] architektura szeregowa jest wykrzystywana w zadaniach związanych z planowaniem i realizowaniem ścieżek robotów. Jednakże ten rodzaj architektur nie zyskał większej popularności z powodu długiego czasu reakcji.
W architekturach równoległych każda warstwa generuje sygnał wyjściowy, nazywany jako *zachowanie*. Moduł decyzyjny określa, które zachowanie ma być realizowane. Dlatego często tego typu rozwiązania są nazywane jako architektury behavioralne. Zespół składający się z jednakowych (homogenicznych) robotów, mający za zadanie przeszukiwanie i gromadzenie się w zadanym miejscu został opisany w pracach [7, 8]. W pracy [47] przedstawiono przykład architektury behavioralnej, w której moduł decyzyjny wybiera takie zachowanie, które jest użyteczne dla całego zespołu. Rozważania na temat aktywacji określonego zachowania, uwzględniające cele także innych robotów są przedstawione w [28]. Prowadzenie robota w formacji jest zaprezentowane w [11]. Sterowanie robotem w bardzo dużej grupie (*ang. VLSR*) z wykorzystaniem sił potencjałowych pomiędzy robotami opisano w [12, 89].

1.3.5 Inne rodzaje architektur

W pracach [22, 23] zwrócono uwagę na trudności w implementacji istniejących modeli formalnych w praktycznych realizacjach systemów wieloagentowych. Według autorów badacze pracujący nad systemami wieloagentowymi dzielą się na dwie grupy: tych, którzy opisują pewne właściwości systemów wieloagentowych za pomocą precyzyjnych modeli formalnych, oraz tych, którzy tworzą systemy wieloagentowe nie wykorzystując żadnych modeli teoretycznych, przez co nie zwracają uwagi na stabilność systemu, sprawność, itp. W związku z tym podjęto próbę takiego opisu systemów wieloagentowych, który mógłby

być wykorzystany w praktyce. Zaproponowano architekturę *M-agenta*, która polega na budowaniu przez agenta modelu środowiska *m*, na podstawie obserwacji. Do danego modelu jest tworzona strategia *s*, która generuje nowy model środowiska *m'*. Modele środowiska *m* i *m'* są porównywane i w zależności od stopnia skomplikowania agenta, w wyniku porównania strategia może być zmieniona, może zostać zastosowana kolejna strategia powodująca nowy model środowska *m"*, itp. Wykonanie stategii *s*, powoduje możliwość porównania czy obserwowany model jest zgodny z przewidywanym modelem *m'*, co pozwala na wprowadzenie elementu uczenia. Realizowanie strategii pozwalających na przejście od modelu środowiska *m - m' - m"* nazywane jest planowaniem, co z kolei umożliwia negocjacje. W modelu *M-agenta* wprowadzono pojęcie profili agenta, różnie reagujących na taką samą obserwację środowiska. Profile można porównać z nastrojem człowieka. W zależności od nastroju danego dnia różnie reagujemy na te same odbierane informacje. W zastosowaniu architektury *M-agenta* w robotyce mobilnej, nie ma założenia, że robot ma być agentem. Zakładając, że roboty komunikują się ze sobą wykorzystując cyberprzestrzeń (sieć internetową), rozróżnia się cztery przypadki umiejscowienia agenta:

- Agent rozszerzony - agent umiejscowiony jest w robocie i jednoczeście w cyberprzestrzeni;
- Agent jest umiejscowiony w robocie a jego odbicie znajduje się w cyberprzestrzeni i służy do przeprowadzania symulacji pracy robotów;
- Agent jest robotem. Agent jest umiejscowiony w cyberprzestrzeni a robot lub roboty są tylko narzędziem agenta;
- Grupa agentów jest robotem. Agenci znajdują się w cyberprzestrzeni i grupa agentów steruje robotem lub robotami;

Wykorzystując agenta wieloprofilowego do powyższych przypadków, jeden profil może obserwować stany i reagować w cyberprzestrzeni, a drugi profil może robić to samo w świecie rzeczywistym. Oba profile mogą mieć te same strategie. Jako przykład rzeczywistego systemu opartego na proponowanym modelu teoretycznym przedstawiono system pomocy medycznej pozwalający na wybranie dla każdego pacjenta najlepszego dla niego szpitala.
W pracy [113] przedstawiono mechanizm umożliwiający tworzenie systemów do sterowania dowolną ilością robotów *(MRROC++ - Multi-Robot Research-Oriented Controller)*. Agenci są podzieleni na agentów upostaciowionych (ang. *embodied*), którzy oddziaływują fizycznie na środowisko, oraz agentów cyfrowych nie posiadających efektorów. Rozróżnia się następujące architektury robotów:

- deliberatywne - działające w oparciu o znany model środowiska, na zasadzie: *czuj - planuj - działaj*;

- reaktywne(behawioralne) - nie posiadające modelu świata działające na zasadzie: *czuj - planuj/reaguj* (jednocześnie);
- hybrydowe - dzielące zadania na behawioralne, które obsługuje agent upostaciowiony, oraz deliberatywne, które obsługuje agent wirtualny;

W chwili obecnej trwają prace nad wykorzystaniem wyżej wspomnianego *M-agenta* w roli agenta wirtualnego w strukturze *MRROC++*, w wyniku czego może powstać jeden system wykorzytujący zalety każdego z powyższych.
Dynamiczne tworzenie zespołów agentów w zmieniającym sie środowisku przedstawiono w pracy [5]. Jako cel nadrzędny przyjęto budowę architektury agenta „samolubnego" (*self-interested*), który będzie mógł być zaimplementowany w rozproszonym środowisku jakim jest na przykład internet. Agenci muszą wymieniać się między sobą posiadanymi zasobami. W związku z tym tworzą się grupy, w których agenci wzajemnie przekazują sobie posiadane zasoby realizując w ten sposób swoje cele. Ponieważ cele agentów mogą być wzajemnie sprzeczne (kilku agentów chce posiadać ten sam zasób), agenci mogą być dołączani lub odłączani od zespołu. Zaproponowano dwa algorytmy, w których agenci byli „*samolubni*", to znaczy nie oddawali swoich zasobów do momentu w którym nie byli członkami zespołu, w którym znajdowały się potrzebne im zasoby. W pierwszym (*simple algorithm*) grupy agentów łączą się z inną grupą, która posiada pożądane zasoby (dla uproszczenia pojedynczego agenta też nazywa się grupą). Agent może być traktowany jako zwykły agent reaktywny, poszerzony o bazę wiedzy, cele, oraz mechanizm kooperacji. Taki algorytm powodował bardzo szybkie zmiany w zespołach, polegające na częstym przyłączaniu i wyrzucaniu agentów z grup. W drugim algorytmie (*electric algorithm*) łączenie się agentów w grupy jest oparte na analizie obwodów elektrycznych. Agenci znajdują się w węzłach, pomiędzy którymi są przewody z rezystorami. „Napięcie" przykładane jest pomiedzy węzłami, w których jeden mogłby przekazać zasób drugiemu. Natężenie prądu w takim grafie pokazuje te węzły (czyli tych agentów), z którymi warto utworzyć grupę, żeby osiągnąć żądany cel. Oczywiście agenci pośredni mają swoje cele i mogą nie być zainteresowani uczestnictwem w takiej grupie. Żeby utworzyć odpowiednią grupę trzeba przeanalizować graf pod kątem celów każdego z agentów. Cechą tego algorytmu było szybkie zwiększanie się grup i wolniejsze ich kurczenie.
Dwa powyższe algorytmy porównano z algorytmem mrówkowym (*ant-hill algorithm*), opartym na znanym z sieci komputerowych statycznym algorytmie rutowania, polegającym na pełnej współpracy wszystkich agentów. Agenci bezinteresownie przekazywali swoje zasoby albo do agentów, które ich potrzebowały, albo do tych agentów, które mogą przybliżyć zasób do agenta docelowego. Porównanie tych algorytmów pokazało, że najlepszy jest pierwszy zaproponowany algorytm, mimo że algorytm mrówkowy jest nieznacznie szybszy. Jednakże w rozproszonym środowisku algorytm mrówkowy nie jest w stanie stwierdzić czy zadanie zostało zakończone. Algorytm „elektryczny" jest zdecydowanie najgorszy. Ogólny wniosek z tych badań mówi, że agent reaktywny poszerzony o bazę wiedzy, cele i mechanizm kooperacji może pracować w dynamicznym środowisku jakim jest internet.

Specyfikację języka serwisów sieciowych *„Entish"* opisującego wymianę informacji pomiędzy agentami, oraz specyfikację protokołu *entish 1.0* służącego do kompozycji serwisów przedstawiono w pracy [4]. Ten mechanizm, podobnie jak SWORD lub XSRL polega na tworzeniu serwisów na bieżąco zgodnie z wymaganiami klientów. Jako przykład systemu wykorzystującego *Entish* przedstawiono agentów softwarowych wyszukujących najlepsze połączenia lotnicze w sieci internet.

1.4 Współpraca w systemach wieloagentowych

Niniejszy podrozdział omawia różne rodzaje współpracy systemów wieloagentowych (SWA). Według [37] system wieloagentowy posiada następujące elementy:

- środowisko;
- obiekty umieszczone w środowisku;
- agenci operujący w środowisku;
- relacji pomiędzy agentami;

Agenci mogą mieć strukturę hierarchiczną, gdzie decyzję podejmuje lider, lub mogą mieć strukturę płaską, (wszyscy członkowie grupy są równorzędni) i przy podejmowaniu wspólnych decyzji konieczny jest mechanizm negocjacji. W strukturze hierarchicznej zaletą jest szybkość podejmowania decyzji, ponieważ nie ma czasochłonnego procesu negocjacji. Wadą systemów hierarchicznych jest niewykorzystanie wszystkich możliwych zasobów grupy. Lider podejmujący decyzję może nie mieć pełnej informacji o otoczeniu, do podjęcia optymalnie słusznej decyzji. W przypadku negocjacji zaletą jest większe prawdopodobieństwo wybrania słusznej decyzji, wadą jest długi czas podejmowania decyzji. Biorąc pod uwagę potwierdzanie odebrania komunikatów, ewentualne powtarzania itp., czas ten może znacznie się wydłużyć. Komunikaty wymieniane pomiędzy agentami mogą być podzielone następująco:

- propozycja określonej akcji;
- akceptacja akcji;
- odrzucenie akcji;
- odwołanie akcji;
- Niezgodzenie się na zaproponowaną akcję;
- Kontrpropozycja na zaproponowaną akcję;

Agent 1 w procesie negocjacji proponuje agentowi 2 podjęcie określonej akcji. Agent 2 może propozycję przyjąć i wysłać potwierdzenie, może propozycji nie przyjąć i wysłać informację do nadawcy o odrzuceniu propozycji, może najpierw propozycję przyjąć a następnie odwołać akcję, oraz może niezgodzić się a nadawcą i wysłać swoją kontrpropozycję. System wieloagentowy posiada następujące cechy:

- środowisko systemu wieloagentowego określa rodzaj komunikacji;
- jest otwarty i podejmowanie decyzji jest rozproszone (nawet w przypadku modelu hierarycznego, każdy agent ma swój poziom autonomii);
- składa się z autonomicznych agentów, z których każdy realizuje swoje cele, które mogą lecz nie muszą być wspólne z interesem grupy;

Arkin [9] dzieli zadania wykonywane przez systemy wieloagentowe na następujące grupy:

- omiatanie otoczenia (sprzątanie, malowanie, odkurzanie, itp);
- przeszukiwanie otoczenia (akcje ratunkowe);
- sterowanie ruchem;
- ruch w formacji;

Jednyn z najnowszych zastosowań systemów wieloagentowych jest wykrzystanie ich w eksploracji przestrzeni kosmicznej [111]. Autorzy jako powód zastosowania SWA podają duże opóźnienia w przesyłaniu sygnału z ziemi, zmniejszenie kosztu, oraz zwiększenie niezawodności systemu. W przeciwieństwie do systemów opierających się na sterowaniu centralnym (np. ASPEN lub Remote Agent System) proponowany system jest całkowicie autonomiczny a planowanie zadań odbywa się pomiędzy agentami planowania i menedżerem planowania. Zadanie jest dekomponowane na plany częściowe, agenci uzgadniają ze sobą ich wykonanie a następnie suma wszystkich planów częściowych daje realizację zadania.

1.4.1 Systemy wieloagentowe a zespoły robotów mobilnych

Systemy wieloagentowe są powszechnie używane w zespołach robotów mobilnych. W niektórych systemach agentem może być każdy czujnik i efektor każdego robota w zespole, chociaż w większości przypadków każdy robot jest oddzielnym agentem. Zadania wykonywane przez zespoły robotów można podzielić na dwie klasy:

1. zadanie może być wykonane przez pojedynczego robota, ale zespół robotów może wykonać je szybciej lub bardziej efektywnie (inspekcja, eksploracja, itp.);
2. zadanie nie może być wykonane przez pojedynczego robota i tylko zespół składający się z kilku robotów współpracujących ze sobą może wykonać zadanie (np. różnego rodzaju zadania transportowe);

Systemy wielorobotowe mają możliwość dekompozycji zadań, co może umożliwić większą prostotę członka zespołu w porównaniu do pojedynczego robota wykonującego to samo zadanie [8]. Inną ważną zaletą takich systemów jest odporność na błędy umożliwiająca wykonanie zadania w przypadku awarii jednego z robotów. Przy wykorzystywaniu zespołu robotów mobilnych trzeba rozwiązać pewne problemy niewystępujące w zadaniu wykonywanym przez pojedynczego robota. Są to: koordynacja pracy, komunikacja, optymalny rozdział zadania, itp. [6, 31].
Zespół robotów może składać się z homogenicznych (*jednakowych*) lub heterogenicznych (*różnych*) robotów. Zespoły robotów homogenicznych mają możliwość redundancji, to znaczy po awarii jednego z robotów inny robot może go zastąpić i zadanie zostanie wykonane. Roboty heterogeniczne mają większe spektrum zastosowań, ponieważ pełnią różne funkcje w zespole. Do wykonania zadania może być konieczna współpraca robotów o różnych właściwościach. W chwili obecnej wydaje się, że ten rodzaj zespołów w przyszłości zyska większą popularność, gdyż część robotów można bogato wyposażać w drogie oprzyrządowanie a pozostałe roboty mogą być dużo prostsze i tańsze co umożliwia masowe ich wykorzystanie i zgodę na ewentualną stratę robota.
Zespoły robotów mobilnych znajdują zastosowanie w:

- **transporcie** - np. ciężkich przedmiotów [6, 63, 49]. W pracach [47, 25, 24] opisano klasyczny przykład zadania wykonywanego przez dwa roboty polegający na przewiezieniu długiej belki przed drzwi, co wymaga współpracy robotów w celu ustawienia się jednego robota za drugim;
- **eksploracji** - czyli zbieraniu informacji o nieznanym otoczeniu [3, 7, 10, 73, 100];
- **inspekcji** - czyli patrolowaniu zadanego obszaru [15, 47, 80, 89]. Przykładem inspekcji mogą być roboty poszukujące miny [104]. Pewnym rodzajem inspekcji może być gromadzenie rozrzuconych elementów w określonym miejscu [79];

Zrealizowanie zespołu robotów mobilnych wykonujących określone zadanie i nie wymagające silnych ograniczeń jest trudne. Dlatego część prac skupia się na zadaniach pośrednich, na przykład na ruchu robotów w szyku [3, 7, 8, 11, 25, 24, 89] lub zawodach polegających najczęściej na grze w piłkę nożną robotów np. [29, 59, 60, 64].

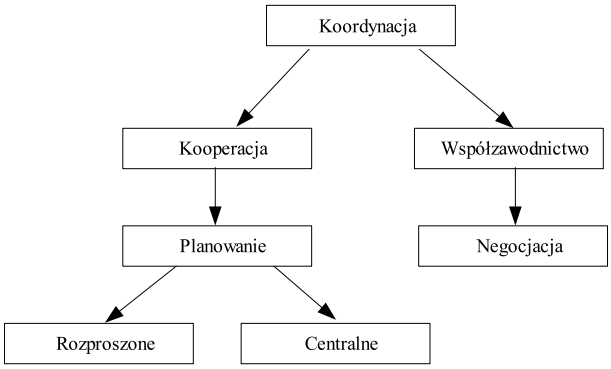
Arkin [9] dzieli zadania wykonywane przez systemy wieloagentowe na następujące grupy:

- omiatanie otoczenia (sprzątanie, malowanie, odkurzanie, itp);
- przeszukiwanie otoczenia (akcje ratunkowe);
- sterowanie ruchem;
- ruch w formacji;

Przykład matematycznego podejścia i omówienie stabilności zespołu w zadaniu polegającym na ruchu po okręgu wokół zadanego punktu (otaczanie) jest przedstawiony w pracy [21].

1.4.2 Komunikacja pomiędzy agentami

W celu omówienia komunikacji zakładamy, że agent ma możliwość odbierania i analizowania danych, oraz wpływania na środowisko. Zakładamy, że agent posiada bazę danych o otoczeniu, oraz posiada możliwość komunikacji (odbierania i wysyłania informacji). Agenci komunikują się ze sobą w celu skoordynowania swojego zachowania, w wyniku czego praca całego systemu jest bardziej spójna. Rodzaje współpracy przedstawia rys.1.13. W przypadku agentów, z własnym celem wewnętrznym agenci przeprowadzają



Rysunek 1.13: Współpraca między agentami.

negocjacje. Koordynacja występuje zawsze gdy w danym środowisku operuje więcej niż jeden agent. W przypadku agentów nieantagonistycznych, pomiędzy agentami zachodzi kooperacja, natomiast współzawodnictwo a w efekcie negocjacja występują pomiędzy agentami, które mają rozbieżne cele (są „samolubne") (rys.1.13). Każdy agent musi posiadać model innych agentów, oraz musi posiadać model komunikacji. System wieloagentowy oceniamy po tym, czy zachowuje się spójnie jako całość. Trudność w realizacji tego zadania polega na tym, że w systemach wieloagentowych przeważnie nie występuje centralne sterownaie. W celu rozwiązania tego problemu agent musi umieć określić wspólne cele, rozwiązywać konfilkty, uaktualniać swoją wiedzę itp. Dlatego użyteczny jest z góry określony model komunikacji pomiędzy agentami. Istniejące modele będą omówione w kolejnych podrozdziałach.
Komunikaty przesyłane pomiędzy agentami możemy podzilić na kilka sposobów. Na przykład na pytania i odpowiedzi. Najprościej zrealizowana komunikacja pomiędzy agentami polega na wysyłaniu informacji przez agentów aktywnych (zdolnych wysyłać i odbierać informacje) do agentów pasywnych (zdolnych tylko do obierania informacji), w postaci rozkazów. Bardziej skomplikowany mechanizm polega na potwierdzaniu otrzymanych rozkazów. Kolejnym krokiem jest dialog. Agent, który odebrał polecenie może

zgodzić się lub odrzucić rozkaz. Podział agentów względem ich możliwości komunikacji przedstawia tab.1.14 [108].

	Agent podstawowy	Agent pasywny	Agent aktywny	Agent równy
Otrzymywanie rozkazów	✔	✔	✔	✔
Otrzymywanie zapytań		✔		✔
Wysyłanie rozkazów		✔	✔	✔
Wysyłanie zapytań			✔	✔

Rysunek 1.14: Podział agentów ze względu na możliwości komunikacyjne.

Protokoły komunikacyjne są składają się z trzech warstw. Najniższa warstwa odpowiada za połączenie, środkowa - za składnię i format informacji. Trzecia warstwa odpowiada za semantykę i typ przesyłanej wiadomości.
Tworząc agentowy model komunikacji warto zwrócić uwagę na sposób komunikowania się między ludźmi. Teoria opisująca to zjawisko nosi nazwę: *"speech act theory"* [98]. Zdania wypowiadane przez ludzi dzielą się na: *prośby, sugestie, zobowiązania i odpowiedzi*. Każda wypowiedź ma 3. aspekty:

- Treść wypowiedzi (np. "przeszukaj pokój P1");
- Intencja wypowiadającego (nadawca chce, żeby pokój P1 został przeszukany);
- Akcja będąca skutkiem wypowiedzi (odbiorca oddala się od szyku w stronę pokoju P1);

Komunikacja między ludźmi często jest niejednoznaczna, zależna od sytuacji, nieprecyzyjna, itd. W świecie sztucznych agentów ilość przesyłanych informacji jest znacznie mniejsza, bardziej precyzyjna i jednoznaczna. świadomość istnienia i znajomość podstaw teorii zajmującej się informacjami wymienianymi między ludźmi, pomaga zdefiniować projektantowi systemów wieloagentowych typy wiadomości, określić skutki przesłania danej informacji, oraz zredukować ilość danych.
Podstawowym zagadnieniem przy tworzeniu modelu komunikacji jest oddzielenie składni protokołu komunikacyjnego (niezależnej od konkretnej implementacji) od semantyki konkretnej informacji (która może zależeć od implementacji). Protokół komunikacyjny powinien być uniwersalny, obsługiwać wszystkie przypadki a jednocześnie zwięzły, zawierający ograniczoną ilość komunikatów. Jednym z takich języków jest KQML (*Knowledge Query and Manipulation Language*)[38]. Jest tojednocześnie język i protokół, w którym komunikaty wymiany danych między agentami, mogą być przesyłane w różnych językach np. KIF, PROLOG, LISP, SQL lub inny.
Częstym problemem jest zbyt duża ilość komunikatów wymienianych pomiędzy agentami, z których zdecydowana większość jest niewykorzystywana. W pracach [16, 48] podjęto

próbę minimalizacji przesyłanych danych. W zadaniu transportowym, zrealizowanym na rzeczywistych robotach, wymiana danych polegała jedynie na wymianie informacji o stanie parametru wewnętrznego robota. Złożoność problemów przy negocjacji pomiędzy agentami softwarowymi jest zaprezentowana w pracy [33].

1.4.3 Protokoły komunikacyjne

Protokoły komunikacyjne służą do wymiany nie tylko pojedynczych komunikatów, ale do prowadzenia konwersacji pomiędzy agentami. W przypadkach, gdy agenci mają rozbieżne cele, lub każdy agent ma swój własny cel, zadaniem protokołu komunikacyjnego jest doprowadzenie do takiej sytuacji, gdy każdy agent będzie mógł spełnić możliwie w jak największym stopniu swoje cele. W przypadkach, gdy agenci mają zbieżne cele, lub gdy jest jeden globalny cel, zadaniem protokołu komunikacyjnego jest doprowadzenie do koherentnej współpracy, bez systemu nadrzędnej kontroli [92].
Protokoły możemy podzielić na: koordynacyjne (*coordination protocols*) i kooperacyjne (*cooperation protocols*), rys.1.13.

Protokoły koordynacyjne

Akcja systemu wieloagentowego musi być koordynowana, gdy system posiada ograniczone zasoby, istnieją zależności pomiędzy agentami i są wspólne ograniczenia. Pojedynczy agent nie ma kompetencji, odpowiednich zasobów i wystarczającej informacji do realizacji zadania. Przykłady koordynacji obejmują: okresową wymianę informacji, sprawdzanie, czy agenci są ze sobą zsynchronizowani, unikanie redundancji, itp. Agent posiada pewien stopień autonomii w generowaniu swoich nowych celów, co powoduje, że agenci mogą nie znać aktualnych celów innych agentów i spójne zachowanie całej grupy jest trudne do uzyskania. Akcje podejmowane przez agenta mogą być opisane w grafie opisującym zależności pomiędzy celami agenta a zasobami potrzebnymi do osiągnięcia poszczególnych celów (liści grafu) (*a*ng. goal graph)[108]. Koordynacja przepływu komunikacji w systemie wieloagentowym obejmuje: *a)* zdefiniowanie grafu z uwzględnieniem zależności między zasobami i celami agentów; *b)* przyporządkowanie części grafu poszczególnym agentom; *c)* sformuowanie warunków, kiedy którą część grafu uaktywniać; *d)* realizację grafu; *e)* badanie czy dobra część grafu jest realizowana. Niektóre z tych punktów są wspólne dla całego systemu, inne - dla pojedynczych agentów.

Protokoły kooperacyjne

Podstawową strategią dla tego typu protokołów jest dekompozycja i dystrybucja zadań. Takie podejście zmniejsza złożność zadania. Mniejsze cele wymagają mniejeszej mocy obliczeniowej agentów i mniejszych zasobów. Przy dekompozycji zadania należy wziąć pod uwagę możliwości agetnów, możliwe konflikty, oraz - jeżeli to możliwe, inne rozwiązanie zadania. Dekompozycji może dokonywać projektant systemu, agent - lider grupy, lub

może ona wynikać z samej definicji zadania. Po dokonaniu dekompozycji może ona być rozesłana do agentów według różnych krytariów [108, 34], np.:

- unikanie przeładowania krytycznych zasobów;
- przydzielanie zadań agentom z dużymi możliwościami;
- utworzenie agenta mogącego samemu przydzielać zadania innym agentom;
- przydzielanie zadań w części pokrywających się z zadaniami innych agentów w celu osiągnięcia koherencji;
- przydzielanie zadań w miarę możliwości od siebie niezależnych. Pozwala to zminimalizować komunikację i synchronizację pomiędzy agentami;
- częstej weryfikacji przydzielonych zadań. Pozwala to rozwiązywać nowe, nagle pojawiające się zadania;

Do rozdziełu zadań służą następujące mechanizmy [108]:

- *Market mechanism* - zadania mają swoją cenę i są przydzielane podobnie jak zakup towarów w sklepie;
- *Contract net* - zapytanie ofertowe, oferta, przetarg;
- *Multiagent planning* - jeden z agentów jest odpowiedzialny za przydział zadań;
- *Organizational structure* - agenci mają z góry narzucone zadania;

Sieć Kontraktowa

Z wyżej wymienionych najbardziej znanym protokołem jest Sieć Kontraktowa (ang. *Contract Net Protocol*). Zasada działania tego protokołu jest podobna do zasady na jakich odbywają się przetargi publiczne [102, 20]. Agent, który chce zlecić wykonanie zadania jest nazywany menedżerem (*manager*), potencjalni wykonawcy - kontrahentami (*contractors*). Protokół z punktu widzenia menedżera jest następujący:

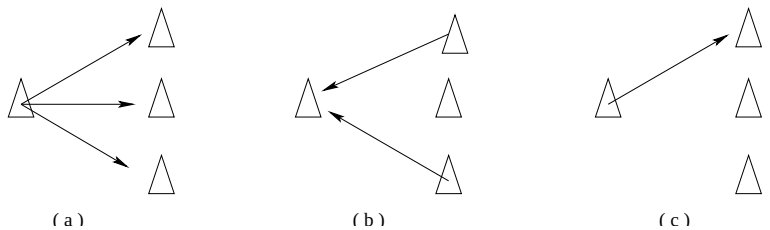
- Wysłanie zapytania ofertowego z zadaniem do wykonania;
- Otrzymanie i analiza ofert od potencjalnych kontrahentów;
- Przyznanie zadania wybranemu kontrahentowi;
- Otrzymanie raportu o wykonaniu zadania.

Z punktu widzenia kontrahenta protokół jest następujący:

- Otrzymanie oferty;

- Analiza możliwości wykonania zadania i zgłoszenie swojej oferty;
- Otrzymanie zlecenia (jeżeli oferta jest najlepsza);
- Wysłanie raportu o wykonaniu zadania;

Schemat działania protokołu *Contract Net* przedstawia rys.1.15.



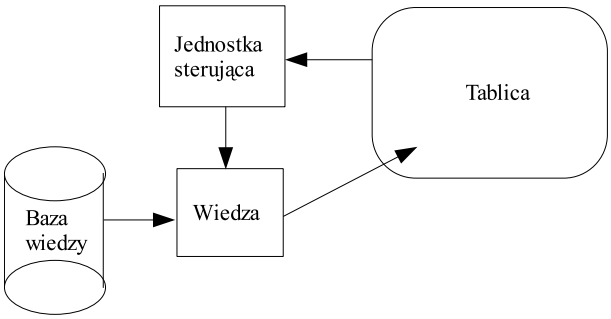
Rysunek 1.15: Schemat zasady działania protokołu *Contract Net*. a) Menedżer wysyła zapytanie ofertowe; b) kontrahenci przysyłają swoje oferty; c) Menedżer wybiera najlepszą ofertę.

Role agentom nie są sztywno przypisane. Każdy agent, który chce żeby wynonano na jego rzecz określoną czynność, może ogłosić przetarg i w ten sposób zostać menedżerem. Zapytanie ofertowe musi zawierać: adresatów (nie wszyscy agenci mogą być zaproszeni do składania ofert), specyfikacje zadania, specyfikacja oferty (opis tego co ma się znaleść w ofercie), oraz termin składnia ofert. Ograniczeniem *Contract Net* jest możliwość przyznania zadania agentowi z mniejszymi możliwościami w przypadku gdy agent bardziej odpowiedni do wykonania zadania nie złoży oferty, lub gdy wykonuje akurat inne zadanie. Drugim ograniczeniem jest to, że menedżer nie ma obowiązku informowania potencjalnych kontrahentów, że przetarg jest już rozstrzygnięty.
Menedżer może nie otrzymać ofert z kilku powodów. Wszyscy potencjalni kontrahenci mogą w danym momencie wykonywać inne zadania, zadanie może być niemożliwe do wykonania przez kontrahentów lub wykonanie zadania może być możliwe, ale nieatrakcyjne dla kontrahentów. Sposobem na rozwiązanie tych problemów może być prośba do kontrahentów o natychmiastową odpowiedź po otrzymaniu oferty, w wypadku nieprzystąpienia do przetargu. Odpowiedź może zawierać jedną z trzech możliwości: **wykonywalne, ale zajęty**, **niewykonywalne**, lub **niezainteresowany**. Dzięki takiej (szybkiej) informacji menedżer może podjąć odpowiednie działanie np. zmodyfikować zadanie, lub poczekać aż kontrahent będzie wolny.
Zaletą *Contract Net Protocol* jest możliwość wyboru z grupy agentów najlepszego, do wykonania konkretnego zadania. Wadą jest stosunkowo długi czas od wysłania ofert do rozstrzygnięcia przetargu.

Blackboard

Istota protokołu wymiany danych typu *Blackboard* polega na istnieniu pamięci wspólnej (tablicy), w której agenci mogą zapisywać i odczytywać dane. Agent, nazywany KS

(*Knowledge Source*), posiadający pewną bazę wiedzy, może w dowonym momencie zapisywać i odczytywać dane z tablicy. W ten sposób każdy agent ma dostęp do danych posiadanych przez innych agentów. Dane są dostępne przez określony czas, lub do momentu nadpisania ich przez dane bardziej aktualne. Schemat architektury typu *Blackboard* przedstawiono na rys. 1.16. Charakterystyka architektury *Blackboard* jest następująca



Rysunek 1.16: Podstawowy schemat architektury typu *Blackboard*.

[108]:

- Każdy agent (KS) jest ekspertem w jednej dziedzinie;
- Wewnętrzna reprezentacja każdego agenta (KSa) jest ukryta;
- Model *blackboard* nie ma wpływu na to, co jest umieszczane w tablicy;
- Każdy agent (KS) musi być w stanie odczytać to, co inny KS zapisał;
- *Blackboard* kontroluje dopływ nowych informacji i usuwanie starych. KS zamiast przeszukiwać całą tablicę może zgłosić pytanie i *blackboard* wyśle odpowiedź;
- W danym momencie tylko jeden agent (KS) może mieć dostęp do tablicy. *Control component* jest mechanizmem, który decyduje, który KS, w danym momencie, może komunikować się z tablicą;
- Każdy agent (KS) może mieć wpływ na rozwiązanie problemu poprzez zaproponowanie nowego rozwiązania, zaprzeczenie istniejącemu, itp.;

1.4.4 Negocjacje

Negocjacja jest procesem, w którym wspólna decyzja jest osiągana przez co najmniej dwóch agentów, z których każdy próbuje osiągnąć indywidualne cele [108]. Agenci najpierw podają swoje stanowiska, które mogą być sprzeczne a następnie próbują osiągnąć wspólne cele. Proces negocjacji ma następujące cechy: język używany przez agenta; protokół negocjacji; kryteria używane przez agentów do zakończenia negocjacji. Istniejące techniki negocjacyjne dzielą się na dwie grupy: **środowiskowe**, oraz **agentowe**.

Techniki środowiskowe skupają się nad następującym problemem: „Jakie powinny być reguły rządzące środowiskiem, żeby agenci, niezależnie od ich intencji, możliwości i oryginalności mogli kooperować efektywnie?" Idealne środowisko powinno mieć następujące cechy: agent nie powinien tracić zasobów na prowadzenie negocjacji(*sprawność*); mechanizm negocjacyjny powinien być tani obliczeniowo(*prostota*); mechanizm negocjacyjny nie powinien wymagać centralnego sterowania(*dystrybucja*); mechanizm powinien równo traktować wszystkich agentów (*symetria*). Dokładny opis tego środowiska można znaleść w [93].
Badacze technik typu agentowego, skupają się nad następującym problemem : „Jaka jest najlepsza strategia w danym środowisku, w którym agent musi kooperować?" Takie podejście zaowocowało opracowaniem wielu strategii dla konkretnych problemów. Trudno określić wspólne elementy, jednak udało się wyodrębnić dwia podstawowe typy. Pierwszy typ polegaja na pełnej formalizacji procesu negocjacji [50]. Drugi typ zakłada, że agenci postępują racjonalnie ekonomicznie. W tym przypadku konieczne są jednak założenia, że grupa składa się z niewielu agentów, mają wspólny język i muszą być w stanie wspólnie rozwiązać dane zadanie. W oparciu o te założenia został opracowany nowy protokół komunikacji: *unified negotiation protocol*. Dokładny opis protokołu jest przedstawiony w [92].

1.5 Teoria podejmowania decyzji

Teoria podejmowania decyzji jest dziedziną, która zajmuje się ustalaniem regół postępowania w celu podejmowania racjonalnych decyzji. Posługuje się modelami decyzyjnymi, obejmującymi funkcję celu, ograniczenia bilansowe oraz warunki brzegowe. W oparciu o warunki brzegowe wyznacza się zmienne decyzyjne, które pozwalaję zmaksymalizować lub zminimalizować funkcjię celu.
Modele decyzyjne dzielą się na:

- modele deterministyczne -rozwiązywane za pomocą analizy matematycznej, programowania liniowego lub nieliniowego (programowanie oznacza schemat działań);
- modele statyczne i probabilistyczne - rozwiązywane za pomocą statystyki matematycznej i rachunku prawdopodobieństwa;
- modele strategiczne - rozwiązywane za pomocą teorii gier;

1.5.1 Teoria gier

Teoria gier bada właściwości gier rozumianych jako procesy o określonych zbiorach strategii. Ważnym elementem w teorii gier jest *suma gry*. Jest to różnica pomiędzy zyskiem wygranego a stratą przegranego. Suma gry może być zerowa lub dowolna. Dodatnia suma gry oznacza że interesy graczy nie wykluczają się wzajemnie. Ze względu na przepływ informacji gry dzielą się na kooperatywne (z istniejącą komunikacją pomiędzy graczmi) i

niekooperatywne (brak komunikacji pomiędzy graczami). Wśród gier niekooperatywnych wyróżniamy gry o niekompletnej informacji, w których uczestnicy nie są pewni realizowanego celu przeciwnika. Ze względu na ilość graczy, gry dzielimy na dwuosobowe i n-osobowe. Ze względu na charakter gry wyróżniamy gry: *normalne(jednokrokowe)* i *ekstensywne(wielokrokowe)*. Gry ekstensywne dzielimy na: *pozycyjne*, *stochastyczne* i *różniczkowe (dynamiczne)*

1.5.2 Gry różniczkowe

Ojcem gier różniczkowych jest Rufus Isaacs. Podstawy matematyczne gier różniczkowych zawarł on w swojej pracy [53]. Gry różniczkowe możemy wykorzystać w trzech klasach problemów sterowania [66]:

- sterowanie obiektem przy braku informacji o zakłóceniach. Zadanie polega na rozwiązaniu gry różniczkowej przy *min max* warunku optymalności. Dane są równania stanu obiektu i zbiór sterowań. Zadanie polega na wyznaczeniu minimum funkcjonału obiektu przy założeniu, że zakłócenie dąży do jego maksimum;

- sterowanie obiektem przy spotkaniu z kilkoma obiektami ruchomymi o różnych wskaźnikach jakości i różnych celach ruchomych.

- synteza wielopoziomowych systemów hierarchicznych (gra różniczkowa może być traktowana jako język hierarchiczny systemów sterowania);

Przykład wykorzystania teorii gier różniczkowych jest przedstawiony w pracy [66]. Autor przedstawia zastosowanie elementów teorii gier do automatyzacji procesu sterowania obiektami ruchomymi na przykładzie podejmowania decyzji manewrowej w nawigacji morskiej. Istnieją następujące rodzaje sterowania ruchem statków dla osiągnięcia określonego celu:

- sterowanie optymalne - podstawowy rodzaj starowania - stabilizacja kursu lub trajektorii;

- gry jednostronne - unikanie kolizji za pomocą manewrów własnego statku, spotkanego statku manewrów kooperujących lub spotkanie statków;

- gry konfliktowe - jednostronne gry dynamiczne (np. brak obserwacji przez jeden ze statków podczas wykonywania manewru przez drugi statek), sytuacja pościgu.

Proces mijania się statków najlepiej opisać za pomocą modelu mijania się statku z j spotkanymi obiektami. Własności procesu opisuje się za pomocą równiań stanu. Synteza sterowania polega na minimalizacji funkcji celu danej w postaci wypłaty całkowej i końcowej. Wypłata całkowa przedstawia straty drogi statku na wymijanie spotkanych

obiektów; wypłata końcowa przedstawia końcowe ryzyko kolizji z j obiektem i końcowe odchylenie statku od zadanej trajektorii [66].

$$I_0^{(j)} = \int_{t_0}^{t_k} [x_0^{(\vartheta_0)}(t)]^2 dt + r_j(t_k) + d(t_k) \rightarrow min.$$

gdzie:
Wypłata całkowa $\int_{t_0}^{t_k} [x_0^{(\vartheta_0)}(t)]^2 dt$ przedstawia straty drogi statku na wymijanie spotkanych obiektów; wypłata końcowa przedstawia końcowe ryzyko kolizji z $r_j(t_k)$ do j obiektu i końcowe odchylenie statku od zadanej trajektorii $d(t_k)$ [66].
Teoria gier wykorzystywana w podejmowaniu decyzji manewrowej w nawigacji morskiej zakłada, że statki są wyposażone w radary umożliwjające śledzenie innych obiektów. W przypadku wyznaczania trajektorii robotów mobilnych, w zadaniach zespołowych istnieje możliwość wykorzystania mechanizmu komunikacji w zespole. W przypadku współzawodnictwa (np. mecze piłki nożnej robotów) lub w zadaniach pościgu ciekawe może być wykorzystanie teorii gier różniczkowych do wyznaczenia optymalnej trasy ruchu robota.

1.6 Metody planowania ścieżki robota

Planowaniu ścieżki robota polega na znalezieniu takiej bezkolizyjnej trasy od bieżącej pozycji robota do punktu docelowego, aby funkcja kosztu przejazdu była jak najmniejsza. Wyróżnia się metody: grafowe, potencjałowe, regułowe i dyfuzyjne [83].

Metody grafowe. Polegają na zbudowaniu grafu, którego węzły powstają w wyniku dyskretyzacji ciągłego obszaru działania robota a krawędzie grafu reprezentują akcje umożliwjające przeprowadzenie robota między tymi punktami. Jeżeli można określić koszt przejścia między punktami reprezentowanymi przez wierzchołki grafu, wówczas do planowania ścieżki wykorzystuje się heurystyczne metody przeszukiwania w grafie ważonym. Wierzchołki grafu wybierane są poprzez rozszerzenie ścian i przeszkód w taki sposób, żeby do dalszej analizy traktować robota jako punkt materialny. Bieżąca pozycja robota również traktowana jest jako wierzchołek grafu.

Metody potencjałowe. Stosowane są w rastrowym modelu środowiska. Robot oraz rastry zajęte przez przeszkody są nośnikami ładunków dodatnich, natomiast cel robota są nośnikami ładunków ujemnych. Rastry wolne posiadają określoną przewodność elektrostatyczną. Taki mechanizm powoduje odpychanie robota od przeszkód, oraz przyciąganie do celu. Ścieżka robota przebiega przez te rastry, dla których suma sił odpychania i przyciągania jest najmniejsza. Poważną wadą tej metody są minima lokalne, występujące przy przeszkodach niewypukłych.

Metody dyfuzyjne. Stosowane są w rastrowym modelu środowiska. Z rastra, w którym znajduje się robot rozchodzi się energia, która jest lekko tłumiona przez rastry wolne,

oraz jest tłumiona do zera przez rastry zajęte przez przeszkody. Po rozejściu się fali, wyznaczenie trasy polega na wyborze od celu do robota kolejnych rastrów z jak najmniejszą wartością [99]. Jednym ze sposobów wykorzystania tej metody jest modelowanie przepływu energii z wykorzystaniem neuronowych sieci komórkowych [BASIA].

Metody regułowe. Przestrzeń poszukiwań jest określana jako dyskretny zbiór stanów robota. Stan jest określany przez prędkość i położenie robota (względem celu i przeszkód). Z każdym stanem związany jest zbiór reguł umożliwjający podjęcie odpowiedniej akcji, przybliżającej robota co celu jak najmniejszym kosztem.

W pracy [26] poszukiwano ścieżek dla robotów pracujących we wspólnym otoczeniu z możliwością częstych kolizji pomiędzy robotami. Przyjęto rastrową reprezentację otoczenia, oraz założono pełną komunikację między agentami. Zaproponowano mechanizm rozpraszania robotów w celu minimalizacji sytuacji konfliktowych.

1.7 Zawody robotów i systemy ratownicze

Zawody robotów

Ogromną popularność w ostatnich latach zdobyły zawody robotów polegające na grze w piłkę nożną. Zawody odbywają się w kilku ligach [90]:

- *liga symulacyjna* - każdy zawodnik jest programem (klientem UDP), który łączy się z serwerem UDP symulującym środowisko (boisko piłkarskie). Każda drużyna składa się z 11 zawodników, z których każdy może być różnym programem. Serwer cyklicznie co 100ms. wysyła komunikaty do klientów z informacją, co zawodnik widzi i słyszy a klienci do serwera mogą wysyłać komunikaty z informacją o wykonywanych czynnościach (bieg, obrót, kopnięcie piłki, krzyknięcie, itp);
- *liga małych robotów* - w drużynie jest 5 robotów o średnicy maksymalnie 18cm, boisko ma rozmiary 280x230cm.;
- *liga średnich robotów* - w drużynie są 4 roboty o średnicy maksymalnie 50cm;
- *liga robotów kroczących* - w drużynie są 4 popularne roboty-pieski Sony Aibo. Boisko ma wymiary 600x400cm. Nie ma sterowania nadrzędnego. Roboty operują całkowicie autonomicznie;
- *liga robotów humanoidalnych* - Rozmiary robotów humanoidalnych są bardzo zróżnicowane: od 10cm. do 2m., dlatego zawody w tej lidze nie są jeszcze rozgrywane, lecz twórcy RoboCup twierdzą, że ta liga jest najbliższa celu jaki sobie wytyczyli: w 2050 roku rozegrać mecz z prawdziwą drużyną piłkarską. Prawdopodobnie w najbliższych latach ta liga będzie silnie rozwijana;

- *liga junior* - zespół składa się z robotów zbudowanych z klocków lego i jest przeznaczona dla młodzieży. Celem istnienia tej ligi są wartości edukacyjne;

Wielu badaczy widzi szerokie pole badań nad sztuczną inteligencją wykorzystując zabawę z grę w piłkę nożną. Otwartym pytaniem pozostaje jak wyniki takich badań naukowych mogą znaleść zastosowanie w innych dziedzinach.
Pomimo, że gra w piłkę robotów jest w pewnym sensie zabawą, po każdych zawodach powstaje wiele poważnych artykułów naukowych na przykład: [77, 82, 58, 14] i wiele innych.

Systemy ratownicze

Twórcami systemu są Hiroaki Kitano (Sony Computer Science Lab, Higashi-Gotanda) i Satoshi Tadokoro (Kobe University), którzy na konferecji ICRA98 w Leuven (Belgia) dyskutowali na temat rozpoczęcia prac nad systemem ratowniczym. Nazwa została zaczerpnięta z dobrze przyjętego systemu RoboCup i dlatego projektowi nadano nazwę: *RoboCupRescue* [91]. Inspiracją było tragiczne trzęsienie ziemi w Kobe (Japonia) w 1995 roku, w którym było ponad 6 tys. zabitych a 80 proc. budynków półtoramilionowego miasta nie nadawało się do dalszej eksploatacji. Katastrofa miała tak tragiczne następstwa ponieważ: *(1)*Trzęsienie ziemi było niespotykanie silne; *(2)*Zniszczone zostały ośrodki udzielania pierwszej pomocy i inne służby miejskie; *(3)*Została zerwana łączność; *(4)*Ośrodki koordynujące akcję ratowniczą a także ludność cywilna nie posiadały dostatecznej informacji o szczegółach wydarzenia;
W związku z powyższym system *RoboCupRescue* z założenia skupia się na:

- zbieraniu, gromadzeniu, sprawdzaniu , analizie, wnioskowaniu i dystrybucji informacji;
- wspomaganiu procesów decyzyjnych;
- testowaniu strategii ratowniczych;
- koordynacji pracy zespołowej;

Celem RoboCupRescue jest aby w 2050 roku w przypadku katastrofy zminimalizować ryzyko służb ratowniczych i zwiększyć szanse na przeżycie ofiar. Umożliwić to ma zastosowanie zespołu autonomicznych robotów, które wykorzystując mechanizm negocjacji będą w stanie:

- znaleść ofiary wypadku i określić stan ich zdrowia;
- dokonać samolokalizacji;
- dostarczyć wyżywienie i łączność;
- określić ryzyko;

- posłużyć jako podpora budowlana;

W chwili obecnej system RoboCupRescue składa się z dwóch lig: symulacyjnej i robotów ratowniczych.

Liga symulacyjna. Podobna jest do do symulacyjnej ligi robocup. Symulowana jest katastrofa (np. trzęsienie ziemi). Uczestniczący w niej agenci mogą być np. strażakami, ofiarami, ochotnikami, przechodniami, itp. Przy modelowaniu zjawiska należy wziąć pod uwagę: planowanie wieloagentowe, różnorodność (heterogeniczmość agentów), podejmowanie decyzji w czasie rzeczywistym, itp.

Liga robotów ratowniczych. Sceną jest makieta ze zburzonym budynkiem. Sparwdzane są skuteczność, dokładność i efektywność znajdowania ofiar imitowanych przez lalki;

Rozdział 2

Cel, założenia i teza pracy

Cel. Celem niniejszej pracy jest poszukiwanie minimalnego zestawu środków zapewniających zespołową pracę robotów ratowniczo-inspekcyjnych.
Jako zadanie do rozwiązania przyjęto inspekcję zadanego obszaru i poszukiwanie pewnych wyróżnionych obiektów przez zespół jednorodnych robotów współpracujących ze sobą. Rozwiązaniem zadania jest zaproponowanie takiego rodzaju współpracy, że zespół robotów wykonuje daną klasę zadań bardziej efektywnie niż pojedynczy robot.

Założenia. Założenia systemu są następujące:

- zespół składa się z grupy jednorodnych robotów, bez lidera;
- system jest całkowicie rozposzony i nie posiada jednoski centralnej;
- mapa jest znana i podzielona na pomieszczenia, które są połączone ze sobą drzwiami. Roboty przed rozpoczęciem zadania znają współrzędne wszystkich ścian i przeszkód w obrębie każdego pomieszczenia, oraz numery drzwi i pomieszczeń. Każde drzwi łączą ze sobą dwa pomieszczenia. Ten mechanizm pozwala robotom planować trasy pomiędzy pomieszczeniami;
- roboty rozpoczynają wykonywanie zadania w jednym miejscu a kończą po przeszukaniu całego obszaru, przy czym pozycja robotów na końcu wykonywania zadania jest nieistotna;
- roboty mają zapewnioną komunikację pomiędzy sobą;
- roboty są wyposażone w czujniki rozpoznające:
  - poszukiwane obiekty;
  - inne roboty;
  - dowolne przeszkody;

- roboty znają swoje położenie;
- celem grupy jest znalezienie pewnych wyróżnionych obiektów, w jak najkrótszym czasie;
- poszukiwane obiekty występują w grupach i nie zmieniają swojego położenia w czasie;
- poszukiwane obiekty posiadają indywidualne identyfikatory, wyróżniające je wzajemnie od siebie;
- rozmiary mapy są przedstawione w pewnych jednostkach długości. Jednostką miary pracy robota jest *krok*. Jeden *krok* jest to przejechanie przez robota jednej jednostki długości;
- z każdym poszukiwanym obiektem związany jest czas, który robot musi poświęcić na „obsłużenie" obiektu. Jeżeli wyobrazimy sobie, że poszukiwanym obiektem jest ranny człowiek, to przy każdym znalezionym rannym robot musi spędzić pewien czas (i w związku z tym wydatkować pewną energię) na niezbędne czynności. Dla każdego rannego człowieka czas, który robot musi przy nim spędzić, jest inny. Dla każdego obiektu czas jest przypisywany losowo przy każdorazowym starcie systemu;
- robot może znajdować się w trybie *dojazdu do pomieszczenia* i *przeszukiwania pomieszczenia*. Jeżeli robot jest w trybie *dojazdu do pomieszczenia* to sensory robota rozpoznające poszukiwane obiekty są wyłączone. Tą sytuację można porównać do samochodu służącego do zamiatania ulic, który jedzie z podniesionymi szczotkami;

Znalezienie poszukiwanego obiektu, uruchamia proces negocjacji, w trakcie której roboty uzgadniają, które z nich przerwą swoje działanie i pojadą przeszukiwać pomieszczenie ze znalezionym obiektem. Roboty obliczają koszt dojazdu do pomieszczenia z obiektami i na tej podstawie podejmują decyzję, czy są zainteresowane przerwaniem swojej bieżącej akcji i dojazdem do pomieszczenia z obiektami. Jeżeli tak, uczestniczą w procesie negocjacji, w trakcie którego zostają wyłonieni zwycięzcy. Tak opisany system można w pełni traktować jako system wieloagentowy. Stąd też w dalszej części pracy pojęcia robot i agent są używane zamiennie.
System może symbolizować zespół ratowniczy, który w budynku ma za zadanie przeszukać budynek. Poszukiwane obiekty mogą być bombami, rannymi ludźmi, lub dowolnymi innymi obiektami, których znalezienie w jak najkrótszym czasie może mieć istotne znaczenie.

Teza pracy. Zastosowanie zaawansowanego mechanizmu negocjacji, wykorzystywanego w systemach wieloagentowych, do zespołu wielu robotów, w pewnej klasie zadań, przyspiesza wykonanie zadania przez zespół robotów, w stopniu większym niż samo zastosowanie grupy wielu robotów wykorzystującej tylko prostą komunikację.

Rozdział 3

Koncepcja zespołu ratowniczego-inspekcyjnego

W poniższych podrozdziałach zostanie opisany schemat zachowania robotów w poszczególnych przypadkach:

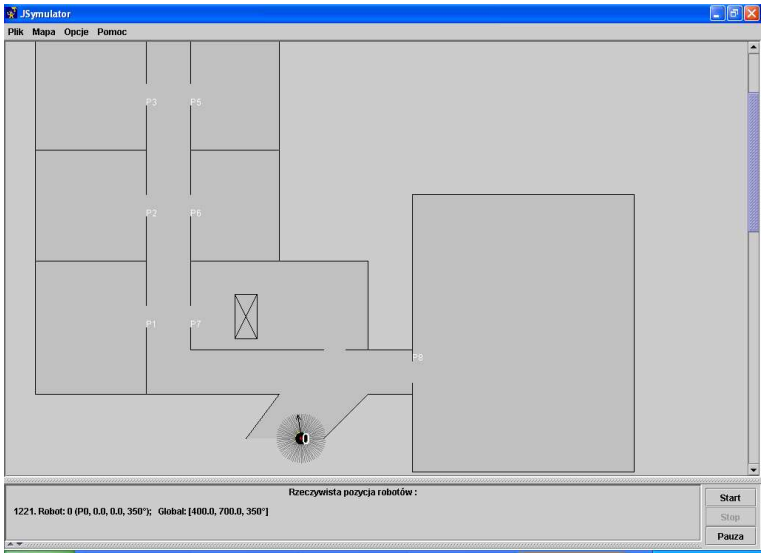
- pojedynczy robot - mapa bez obiektów;
- pojedynczy robot - mapa z obiektami;
- zespół wielu robotów - mapa bez obiektów;
- zespół wielu robotów - mapa z obiektami;

Najbardziej zaawansowany jest ostatni zespół wielu robotów znajdujący się w środowisku z obiektami. Jednak w celu szczegółowego opisania całości zagadnienia zaprezentowano kolejno wszystkie przypadki.

3.1 Jeden robot w budynku bez poszukiwanych obiektów

Rysunek 3.1 przedstawia robota w budynku, który ma być przeszukany. Koło wokół robota przedstawia zasięg sensorów robota. Budynek jest podzielony na pomieszczenia, które są połączone między sobą drzwiami. Mapa i rozmiary robota są przedstawione we własnych jednostkach długości. Wprowadzono pojęcie **„krok"** robota, które oznacza przejechanie przez robota jednej jednostki długości. Koszt pracy robota oznacza wykonanie określonej ilości kroków. Wykonanie przez robota obrotu o dowolny kąt nie jest trakowane jako koszt.

Ogólna koncepcja przeszukania pomieszczenia polega na dojechaniu i sprawdzeniu każdego pomieszczenia. Dojazd do pomieszczenia odbywa się z wykorzystaniem metod



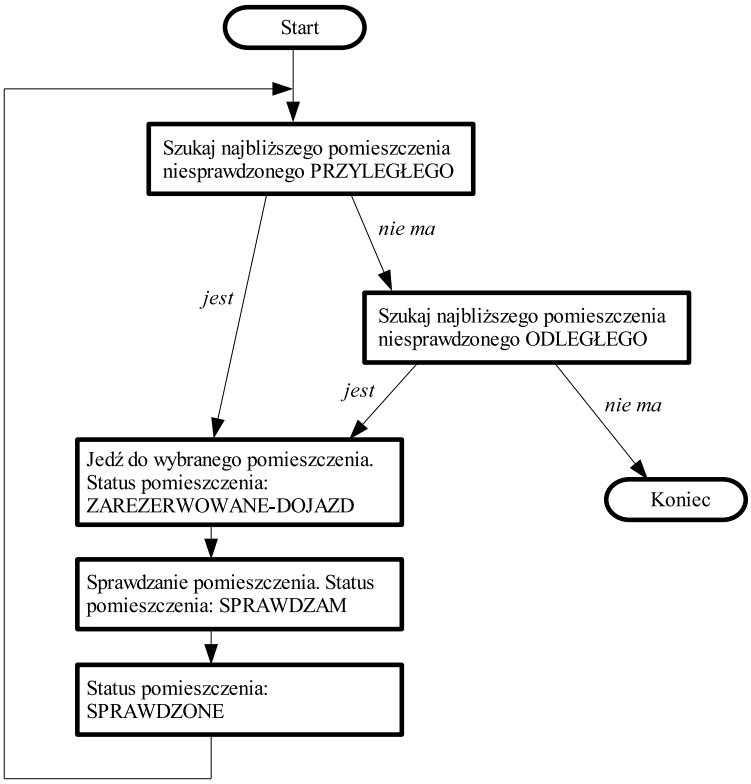
Rysunek 3.1: Robot w budynku, który ma być przeszukany

grafowych (przeszukiwanie grafu "w głąb", algorytm A*); sprawdzanie pojedynczego pomieszczenia odbywa się z wykorzystaniem metod rastrowych;
Pomieszczenia mogą znajdować się w jednym ze stanów: *niesprawdzone, zarezerwowane-dojazd, sprawdzane, sprawdzone*. Tryb *zarezerwowane-dojazd*, oznacza, że robot wybrał to pomieszczenie jako najbliższe i jest w trakcie dojazdu. Ten tryb jest różny od trybu *zarezerwowane*, który będzie występował w kolejnych podrozdziałach. Pozostałe tryby są oczywiste i nie wymagają komentarza.
Robot, wybierając pomieszczenie, do którego chce dojechać rozróżnia pomieszczenia: *przyległe* i *odległe*. Pomieszczenie *przyległe* graniczy z danym pomieszczeniem, natomiast do pomieszczenia *odległego* trzeba jechać przez pomieszczenia pośrednie.

Dla mapy z rys. 3.1 algorym wykonania zadania może wykonać jak na rys. 3.2.

Dla omawianego przypadku kolejność sprawdzania pomieszczeń może być następująca: **P0 - P7 - P1 - P2 - P3 - P4 - P5 - P6 - P8** przy założeniu, że robot skończył sprawdzanie pomieszczenia P0 w okolicy pomieszczenia P7, oraz skończył sprawdzanie pomieszczenia P7 w okolicy drzwi prowadzących do pomieszczenia P1 (z lewej strony). W przypadku, gdy robot skończył sprawdzanie pomieszczenia P0 w przeciwległym końcu niż rozpoczął przeszukiwanie kolejność sprawdzania pomieszczeń mogła by być następująca **P0 - P4 - P5 - P6 - P7 - P8 - P1 - P2 - P3**. Różne wyniki są spowodowane pewną losowością przy sprawdzaniu pomieszczeń (losowy kąt obrotu przy wjeździe do pomieszczenia).
Robot po rozpoczęciu zadania wykonuje następujące etapy:

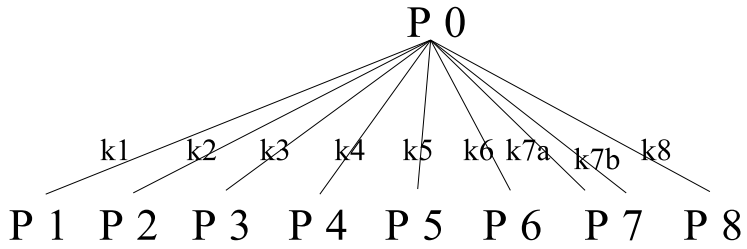
1. **Wczytanie mapy.** Robot wczytuje dane: ściany, drzwi i znane przeszkody dla każdego pomieszczenia. Następnie tworzone są trzy rodzaje grafów:



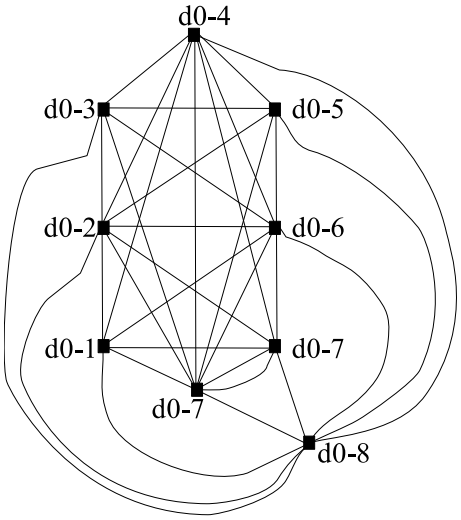
Rysunek 3.2: Robot w trakcie przeszukiwania pomieszczenia

- graf całego budynku zawierający w węzłach pomieszczenia, przedstawiony w postaci drzewa (rys.3.3);
- graf całego budynku zawierający w węzłach drzwi (rys.3.4);
- grafy dla każdego pomieszczenia, zawierające w węzłach: punkty przecięcia linii rozszerzających wszystkie przeszkody (jeżeli znajdują się wewnątrz pomieszczenia), środki drzwi, oraz bieżącą pozycję robota (jeżeli robot w znajduje się w pomieszczeniu) (rys.3.5). Taki graf pozwala na bezkolizyjne poruszanie się robota od bieżącego położenia do dowolnych drzwi;

2. **Wybór pomieszczenia.** Graf, mający pomieszczenia w węzłach, zostaje przedstawiony jako drzewo, którego korzeniem jest pomieszczenie, w którym znajduje się robot (rys.3.3). Za pomocą algorytmu „w głąb" robot znajduje pomieszczenie do sprawdzenia. W pierwszym kroku robot wybierze pomieszczenie, w którym robot znajduje się (czyli korzeń drzewa). Przy wyborze pomieszczenia do sprawdzenia naistotniejsza jest odległość od bieżącej pozycji robota, do każdego pomieszczenia (czyli koszt dojazdu do najbliższych drzwi prowadzących do pomieszczenia). W dalszej prezentacji systemu przyjęto, że robot zakończył sprawdzanie pomieszczenia P0 w miejscu, w którym zaczął. Dla mapy z rys.3.1 koszty dojazdu do poszczegól-



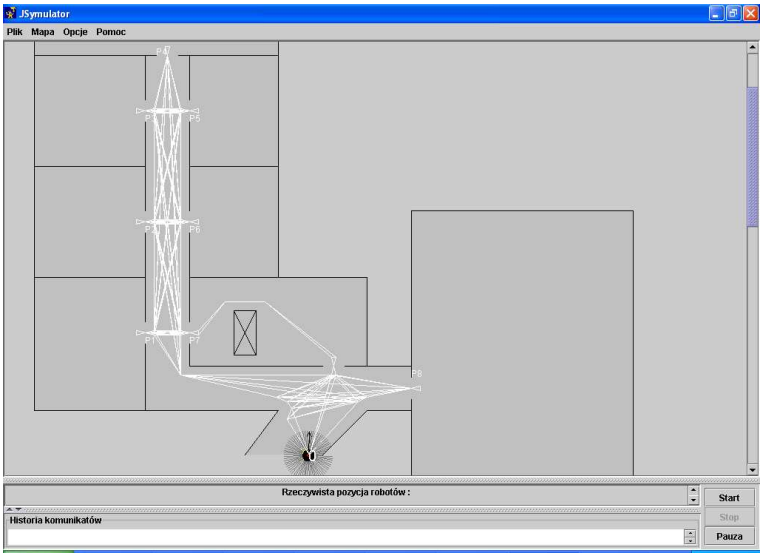
Rysunek 3.3: Graf mapy z rys. 3.1, z pomieszczeniami w węzłach, w formie drzewa; k-koszt przejazdu pomiędzy pomieszczeniami;



Rysunek 3.4: Graf mapy z rys. 3.1, z drzwiami w węzłach. Ze względu na czytelność rysunku pominięto przedstawienie kosztu przejazdu pomiędzy każdymi drzwiami;

nych pomieszczeń przedstawione są w tablicy 3.1. Z tabeli widać, że najbliższe jest pomieszczenie P7 (koszt dojazdu wynosi: 43 kroki). Robot oznacza to pomieszczenie jako *zarezerwowane-dojazd*, po czym jedzie do tego pomieszczenia. Z pomieszczenia P0 do P7 prowadzi dwoje drzwi. Robot rozpatruje drogę przez oboje drzwi, po czym do dalszych obliczeń uwzględnia tylko drogę krótszą. Mechanizm dojazdu do pomieszczenia jest przedstawiony w dalszej części podrozdziału.

3. **Dojazd do pomieszczenia** Robot w trakcie dojazdu do pomieszczenie porusza się po grafie zbudowanym poprzez rozszerzenie ścian i przeszkód. Cała droga jest zapisana jako zestaw par: *(obrót, jazda)*. Zestawy par podzielone są na grupy: od bieżącej pozycji do najbliższych drzwi, a następnie od drzwi, do następnych drzwi, itd. aż do osiągnięcia drzwi docelowych. Droga od punktu początkowego do pomieszczenia P7 jest zapisana jako para (21, 43) (rys.3.1). Oznacza, to że żeby dojechać do pomieszczenia P7 robot musi wykonać obrót o 21^0 i przejechać



Rysunek 3.5: Rozszerzenie ścian i przeszkód w celu utworzenia grafu

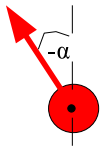
z pom. P0 do pom.:	koszt dojazdu
$P1$	100
$P2$	147
$P3$	197
$P4$	218
$P5$	196
$P6$	146
$P7$	43
$P8$	59

Tablica 3.1: Koszt dojazdu z początkowej pozycji robota w pomieszczeniu P0, do każdego z innych pomieszczeń

prosto odcinek o długości 43 kroków. Kąt obrotu oznacza odchylenie od linii pionowej (rys.3.6). Koszt wykonania tego przejazdu wynosi 43 co jest zgodne z tablicą 3.1 (obrotu nie wlicza się do kosztu). Inny przykład przedstawia przejazd z punktu początkowego do pomieszczenia P1, tablica 3.2. Graficznie droga dojazdu do pomieszczenia P1 jest przedstawiona na rys. 3.7.

Korekcja ścieżki w trakcie dojazdu do pomieszczenia

W trakcie dojazdu do pomieszczenia, ścieżka zawsze prowadzi przez przynajmniej jedne drzwi. Robot przejeżdzając przez wszystkie mijane drzwi weryfikuje swoje położenie. Jeżeli okaże się, że robot nie znajduje się w zakładanym punkcie,

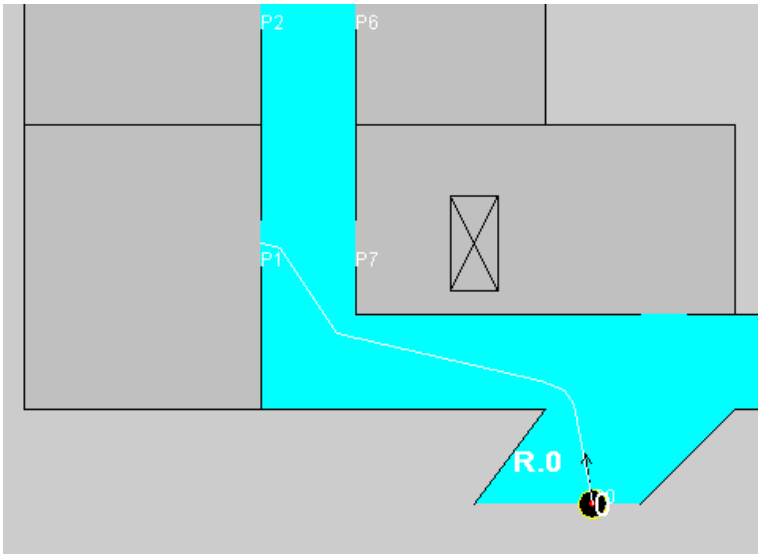


Rysunek 3.6: Obrót robota o kąt ujemny;

obrót o kąt	*jazda prosto*
-11	21
-34	4
-68	21
-77	44
-34	22
-76	4

Tablica 3.2: Dojazd robota z pozycji początkowej w pomieszczeniu P0 do pomieszczenia P1. Graficznie dojazd jest przedstawiony na rys. 3.7

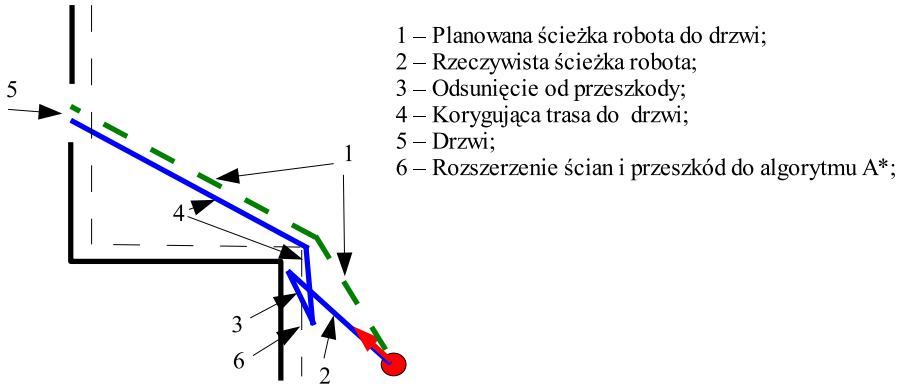
.



Rysunek 3.7: Droga robota od punktu początkowego do pomieszczenia P1

wprowadzana jest ścieżka korygująca od bieżącej pozycji robota, do środka drzwi. Taką realizację ścieżki można porównać do statku na morzu, który płynie od jednej latarni morskiej do następnej. Przyczyny błędnej realizacji ścieżki mogą być spowodowane albo wykryciem przeszkody przed sensory robota, albo zaokrągleniem kąta obrotu do pełnych stopni. Jeżeli sensory robota wykryją przeszkodę, robot nie

jedzie do przodu, lecz wykonuje krok „w miejscu". Ten sposób pozwala na zakończenie realizacji ścieżki, co umożliwia wprowadzenie korekcji.
Korekcja ścieżki polega na odsunięciu się od przeszkody i ponownym obliczeniu trasy od bieżącej pozycji, do punktu pośredniego w drzwiach. Odsunięcie od przeszkody jest konieczne, ponieważ ścieżka jest poszukiwana za pomocą algorytmu A* i znalezienie ścieżki od punktu będącego bliżej przeszkody niż linia rozszerzająca przeszkody jest niemożliwe. Sytuacja, gdy robot nie dojechał do drzwi i wprowadził ścieżkę korygującą przedstawiona jest na rys. 3.8.



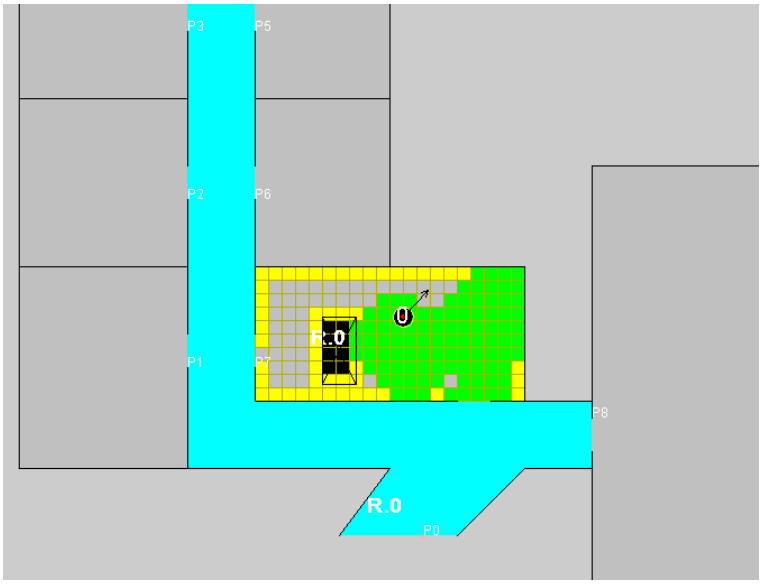
Rysunek 3.8: Wprowadzenie ścieżki korygującej w trakcie dojazdu do pomieszczenia

Omijanie przeszkód

W momencie, gdy przeszkoda uniemożliwi robotowi realizację ścieżki, robot wykonuje operację omijania przeszkody. Polega ona na obrocie o $+30^0$, lub -30^0 i jeździe o losową wartość w przedziale od 5 do 15 kroków w przód lub w tył, w zależności od wskazań sensorów. Te wartości zostały dobrane empirycznie i pozwalają uwolnić się robotowi z sytuacji zakleszczenia (sytuacji, gdy robot nie jest w stanie wykonać żadnego kroku). Operacja omijania przeszkody została przedstawiona w poprzednim podpunkcie jako odsunięcie się od ściany (nr 3 na rys.3.8).

4. **Przeszukanie pomieszczenia.** Po dojechaniu do wybranego pomieszczenia, robot rozpoczyna pracę w trybie rastrowym. Rastry w pomieszczeniu podzielone są na grupy: *niesprawdzony*, *sprawdzony*, *zajęty* (w części rastra znajduje się ściana lub przeszkoda i wjechanie do rastra całego robota może nie być możliwe), *niedostępny* (raster znajduje się wewnątrz przeszkody i jego sprawdzenie jest niemożliwe). Pomieszczenie uznaje się za sprawdzone, jeżeli nie ma żadnych niesprawdzonych rastrów. Robot w trakcie sprawdzania pomieszczenia jest przedstawiony na rysunku 3.9. Po wjechaniu do pomieszczenia, robot obraca się o losowy kąt z zakresu -90 - +90 stopni (żeby nie wyjechać z pomieszczenia), po czym sprawdza pomieszczenie jadąc prosto

aż do napotkania ściany. Po dojechaniu do ściany robot obraca się na zasadzie odbijającej się piłki (obrót o kąt symetryczny do normalnej). W każdym kroku robota sprawdzane są rastry znajdujące się w promieniu „widzenia" robota. W momencie gdy 90% pomieszczenia zostanie sprawdzone, robot przerywa sprawdzanie ruchem chaotycznym i jedzie do najbliższego niesprawdzonego rastra. W ten sposób robot sprawdza ostatnie 10% pomieszczenia. W przypadku, gdy robot nie sprawdził jeszcze 90% ale w ciągu ostatnich 5 kroków nie sprawdził żadnego rastra, robot przerywa sprawdzanie ruchem chaotycznym i jedzie do najbliżdzego niesprawdzonego rastra. Po sprawdzeniu niesprawdzonego rastra, ruch „chaotyczny" jest kontynuowany. Wartości 90% i 5 ostatnich kroków zostały dobrane empirycznie i być może dobór innych wartości dałby lepsze wyniki. Jeżeli zdarzy się, że robot dojedzie do przeszkody pod kątem prostym, obraca się o kąt losowy. Mechanizm wyboru wolnego rastra i dojazd do niego jest przedstawiony w dalszej części podrozdziału.



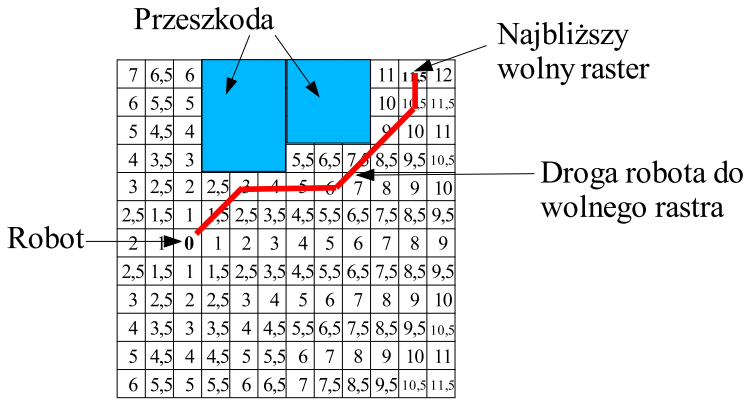
Rysunek 3.9: Robot w trakcie przeszukiwania pomieszczenia

Po zakończeniu przeszukiwania pomieszczenia P7 robot ponownie sprawdzi koszt połączeń do wszystkich niesprawdzonych pomieszczeń i wybierze najbliższe pomieszczenie (porównywalnie do tablicy 3.1), dojedzie do tego pomieszczenia i sprawdzi je. Taki cykl będzie trwał aż do momentu gdy wszystkie pomieszczenia będą sprawdzone.

Wybór i dojazd do niesprawdzonego rastra

Jeżeli robot jest w trybie sprawdzania pomieszczenia i albo sprawdził 90% pomieszczenia, albo w okolicy robota nie ma niesprawdzonych rastrów (nie sprawdził żadnego rastra w ciągu ostatnich 5. kroków), to robot wybiera najbliższy niesprawdzony raster po czym

jedzie do niego. Wybór najbliższego niesprawdzonego rastra odbywa się z wykorzystaniem dyfuzji. Przyjęto że raster, w którym znajduje się robot ma wartość 0 a każdy następny raster w pionie i w poziomie zwiększa swoją wartość o 1, natomiast sąsiednie po przekątnej zwiększają wartość o 1,5, przy czym zawsze dla rastra wybiera się wartość najmniejszą z możliwych. Każdy raster „pamięta", który raster przypisał mu wartość. Po sprawdzeniu wszystkich rastrów, robot wybiera niesprawdzony raster z najmniejszą wartością. Pamiętając swojego poprzednika, wiadomo, przez które rastry robot ma dojechać do niesprawdzonego rastra. Przykład wartości przypisanym rastrom przedstawia rysunek 3.10.

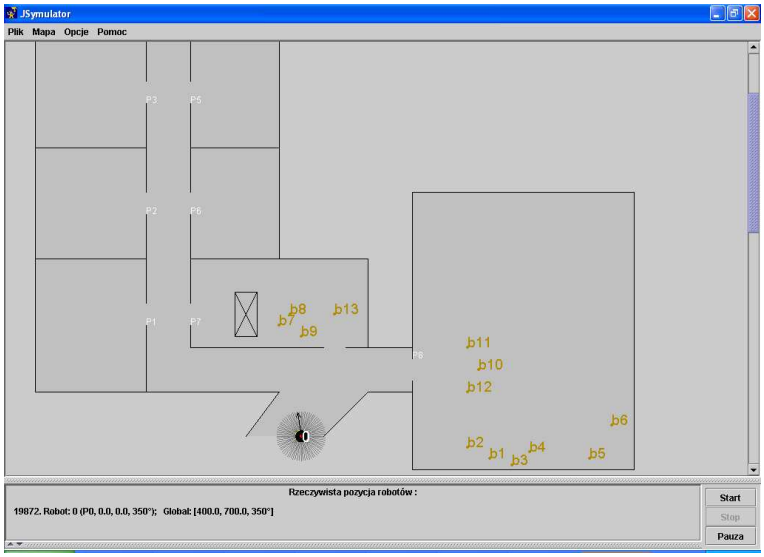


Rysunek 3.10: Droga robota do najbliższego wolnego rastra znaleziona za pomocą dyfuzji

3.2 Jeden robot w budynku z poszukiwanymi obiektami

W poprzednim podrozdziale opisano zachowanie robota przy przeszukiwaniu zadanego obszaru, w którym nie występują poszukiwane obiekty. Obecnie rozszerzamy zachowanie robota o przypadek, w którym robot znajdzie poszukiwany obiekt. Z każdym obiektem związana jest pewna wartość oznaczająca ile czasu (a ściśle ile „kroków") robot musi poświęcić na obsługę obiektu, zanim będzie mógł kontynuować przeszukiwanie pomieszczenia. Ponieważ w rozpatrywanym przypadku występuje tylko jeden robot, różnica z poprzednim przypadkiem polega tylko na wykryciu obiektu i zatrzymaniu się przy nim w celu „obsłużenia" obiektu. Mapa obszaru, który robot ma sprawdzić wraz z poszukiwanymi obiektami jest przedstawiona na rys.3.11.

Wczytanie mapy, wybór pomieszczenia i dojazd do pomieszczenia są identyczne jak w przypadku bez poszukiwanych obiektów. Różnica pojawia się dopiero w trakcie przeszukiwania pomieszczenia. Jeżeli w trakcie przeszukiwania pomieszczenia robot wykryje obecność poszukiwanego obiektu, zatrzymuje się i „obsługuje" znaleziony obiekt. „Obsługa" polega na odczytaniu liczby związanej z danym obiektem i wykonanie tylu kroków „w



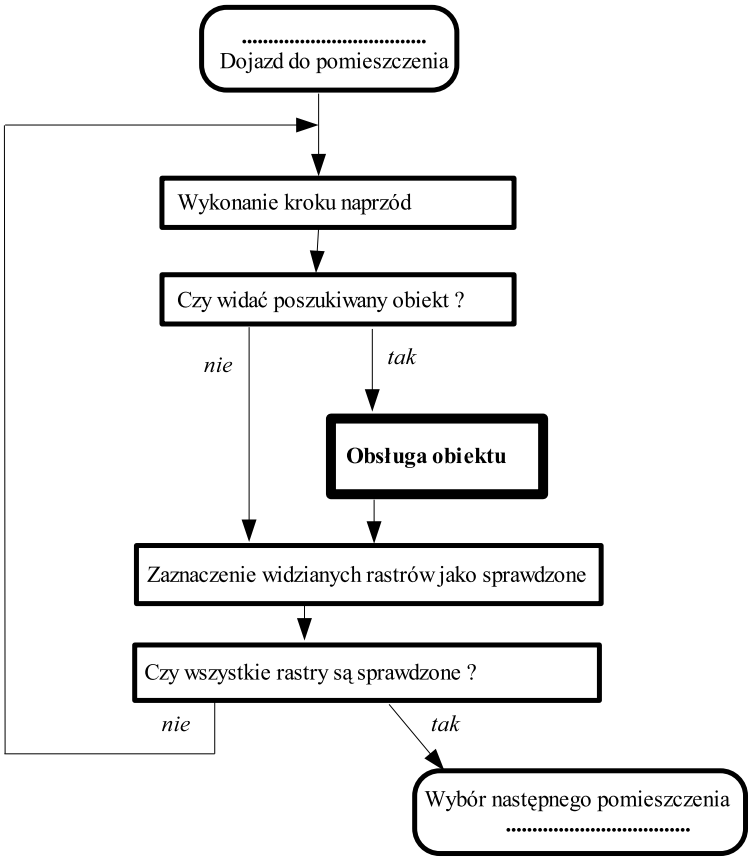
Rysunek 3.11: Poszukiwane obiekty (b1-b13), które mają być znalezione w sprawdzanym obszarze

miejscu" ile wynosi liczba związana z obiektem. Może to symbolizować pomoc rannemu człowiekowi, lub czas potrzebny na rozbrojenie bomby. Schemat przeszukiwania obszaru jest podobny do schematu z rys.3.2, przy czym należy rozbudować blok sprawdzania pomieszczenia. Szczegółowy mechanizm sprawdzania pomieszczenia wraz z obsługą poszukiwanego obiektu przedstawia schemat na rys.3.12.

3.3 Wiele robotów w budynku bez poszukiwanych obiektów

Mechanizm sprawdzania obszaru przez zespół wielu robotów, zasadniczo różni się od zadania wykonywanego przez jednego robota. Główna różnica polega na równomiernym podzieleniu zadania na wszystkie roboty, w taki sposób, żeby roboty sobie nawzajem nie przeszkadzały. Zbyt wiele robotów w bezpośredniej bliskości powoduje spowolnienie wykonania całego zadania. Bardzo dobrym przkładem jest próba przejechania wielu robotów przez drzwi w jednym momencie. Jeżeli roboty potrafią uzgodnić między sobą kolejność przejazdu, wtedy cała operacja wykonuje się stosunkowo sprawnie. Jednakże zawsze istnieje pewna graniczna liczba robotów, której przekroczenie musi spowodować spadek efektywności. Sytuacja, w której zbyt wiele robotów próbuje jednocześnie przejechać przed drzwi jest przedstawiona na rys.3.13. Podstawowe mechanizmy, o które został rozszerzony przypadek wielu robotów, obejmują:

- wybór pomieszczenia do sprawdzenia. Odpowieni wybór pomieszczenia spowoduje,



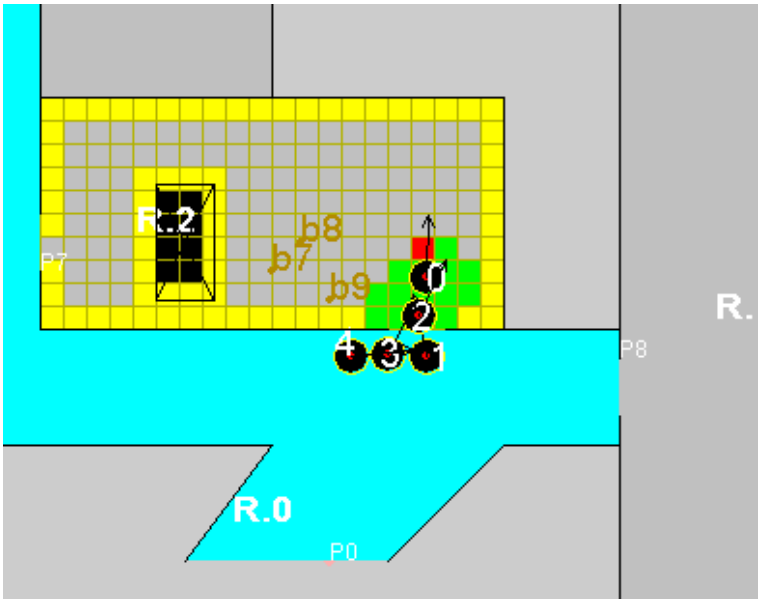
Rysunek 3.12: Schemat przeszukiwania pomieszczenia, wraz z obsługą znalezionego obiektu

że roboty rozproszą się;

- wprowadzenie mechanizmu komunikacji między robotami umożliwiającej podział zadania, oraz unikanie „zakleszczeń" robotów;
- wprowadzenie machanizmu pomocy innym robotom, gdy robot skończył swoje zadanie;

3.3.1 Wybór pomieszczenia do sprawdzenia

W przypadku pojedynczego robota, wybór pomieszczenia do sprawdzenia odbywał się z wykorzystaniem klasycznego algorytmu przeszukiwania grafów „w głąb", gdzie zawsze pierwsze sprawdzane było pomieszczenie, w którym robot rozpoczynał swoje zadanie. Ten system dobry dla pojedynczego robota ma jednak wadę w przypadku większej liczby robotów. Robot rozpoczynający przeszukiwanie pomieszczenia przeszkadza innym robotom w ich poruszaniu się. Taka sytuacja ma miejsce nie tylko w pomieszczeniu, w którym roboty rozpoczynają zadanie, ale może wystąpić, w każdym innym, w którym jeden robot sprawdza pomieszczenie a drugi przez nie przejeżdża. Dlatego w celu zmniejszenia ryzyka tego typu

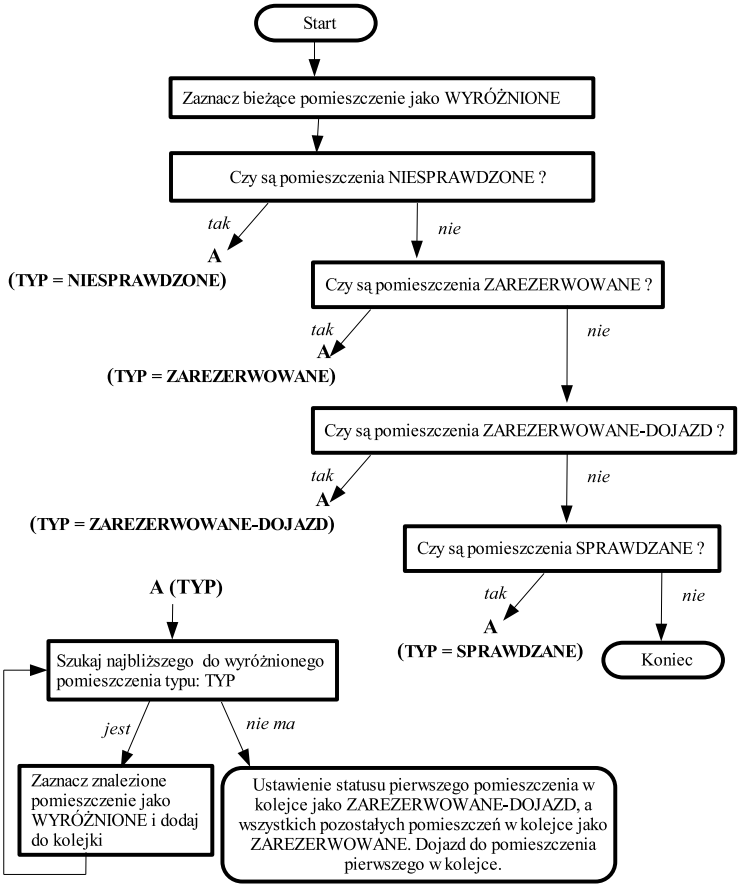


Rysunek 3.13: Zbyt wiele robotów próbujących jednocześnie przejechać przez drzwi

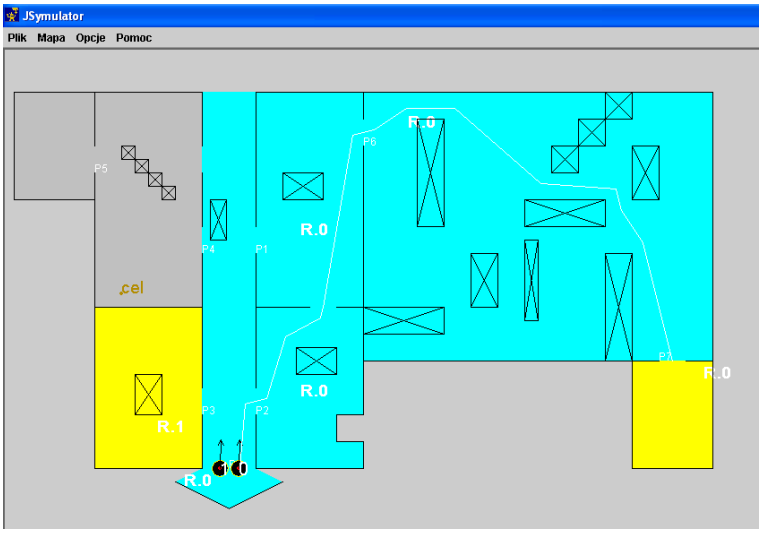
kolizji przy wyborze pomieszczenia algorytm został zmieniony tak, żeby maksymalnie rozporszyć roboty. Ogólna idea polega na tym, aby każdy robot szukając najbliższych wolnych pomieszczeń pojechał do końca gałęzi, a następnie sprawdzał pomieszczenia od końca gałęzi do korzenia. W tym celu wprowadzono nowy stan, w którymy może znajdować się pomieszczenie: *zarezerwowane*. W ten sposób oznaczane są pomieszczenia, które nie leżą na końcu gałęzi. Pomieszczenie na końcu gałęzi jest oznaczane jako *zarezerwowane-dojazd*. Schemat przedstawiający algorytm wyboru pomieszczeń do sprawdzenia, rezerwowaniu ich i zapisywaniu w kolejce pomieszczeń do sprawdzenia przedstawia rysunek 3.14.

Rys.3.15 przedstawia dojazd robotów do pomieszczenia na końcu gałęzi, wraz z zarezerwowaniem wszystkich pomieszczeń pośrednich.

Jedynym przypadkiem, w sytuacji gdy nie występują poszukiwane obiekty, kiedy robot może pojechać do pomieszczenia zarezerwowanego lub sprawdzanego przez innego robota jest sytuacja, w której robot skończył sprawdzanie swojej gałęzi i nie znalazł żadnych niesprawdzonych pomieszczeń. Wtedy robot szuka kolejno pomieszczeń w trybie: *zarezerwowane, zarezerwowane-dojazd, sprawdzane*. Taki mechanizm pozwala na większą efektywność zespołu, gdyż w przeciwnym wypadku robot, który sprawdził krótszą gałąź, musiałby czekać na robota, który sprawdza gałąź dłuższą. W przypadku, gdy dwa lub więcej robotów sprawdza jedno pomieszczenie, roboty w każdym z kroków wysyłają komunikat z informacją, które rastry w danym kroku zostały sprawdzone.



Rysunek 3.14: Schemat wyboru i rezerwacji pomieszczeń do sprawdzenia



Rysunek 3.15: Roboty w trakcie dojazdu do pomieszczeń na końcu gałęzi. Wszystkie pomieszczenia pośrednie są oznaczone jako *zarezerwowane*

3.3.2 Komunikacja między robotami

W celu umożliwienia prawidłowej pracy zespołu wielu robotów, konieczna jest komunikacja pomiędzy robotami. Komunikacja jest potrzebna do:

- podziału zadań;
- unikaniu kolizji;
- wspólnego sprawdzania pomieszczenia;

W zespole nie istnieje wyróżniony lider i w trakcie wykonywania zadania roboty są sobie równorzędne. Ponieważ roboty muszą mieć wspólną wiedzę na temat aktualnego stanu środowiska, konieczny jest mechanizm, który uniemożliwia powstawanie rozbieżności (np. jednoczesna rezerwacja tego samego pomieszczenia). Roboty są ponumerowane i komunikacja odbywa się na zasadzie „przeskakującego żetonu". Roboty kolejno od 0 do największego numeru otrzymują „żeton", wykonują swoje zadanie i przekazują „żeton" do robota o następnym numerze. Technologia ta została oparta na bazie techniki nazywanej *token passing*, wykorzystywanej w topologii pierścieniowej, jedenej z architektur systemów otwartych w sieciach komputerowych [61]. Zadania, które roboty wykonują w trakcie posiadania żetonu polegają na sprawdzaniu lub modyfikowaniu wspólnych danych, czyli: wybór pomieszczenia do sprawdzenia, wysyłanie komunikatów zmieniających status pomieszczeń, sprawdzanie statusu pomieszczeń.

3.3.3 Format komunikatów

Do rozwiązania zadania przeszukiwania budynku bez poszukiwanych obiektów opracowano dwa protokoły komunikacyjne: **kpo** *(kto, pomieszczenie, operacja)* i **kro** *(kto, robot, operacja)*. Pierwszy człon w obu protokołach oznacza nadawcę, czyli numer robota, który wysyła komunikat. Pozostałe człony są różne.

Protokół kpo

Protokół **kpo** służy do:

- zmiany statusu pomieszczeń;
- informowaniu o sprawdzonych rastrach, przy wspólnym sprawdzaniu pomieszczenia;
- przekazywaniu „żetonu" następnemu robotowi;

Status pomieszczenia	*Numer statusu*
Niesprawdzone	20
Zarezerwowane	21
Zarezerwowane-dojazd	22
Sprawdzane	23
Sprawdzony	24

Tablica 3.3: Statusy pomieszczeń i odpowiadająca im wartość liczbowa

.

Zmiana statusu pomieszczeń. Drugi człon mówi, którego pomieszczenia dotyczy komunikat a trzeci człon oznacza jaki jest nowy status pomieszczenia. Na potrzeby komunikacji wszystkie statusy są ponumerowane co przedstawiono w tablicy 3.3.

Na przykład zarezerwowanie i dojazd do pomieszczenia P7 przez robota z numerem 0, powoduje wysłanie komunikatu: <**0.P7.22**>; rozpoczęcie sprawdzania pomieszczenia P8 przez robota nr 1: <**1.P8.23**>; zakończenie sprawdzania pomieszczenia P1 przez robota nr 3: <**3.P1.24**> i tak dalej.

Wspólne sprawdzanie pomieszczenia. W przypadku gdy więcej niż jeden robot sprawdza jedno pomieszczenie, roboty w każdym kroku wymieniają informacje, które rastry w danym kroku są sprawdzone. Pierwsze dwa człony protokołu **kpo** są identyczne jak w poprzednim przypadku. Trzeci człon określający, że paczka dotyczy sprawdzonych rastrów jest oznaczony numerem 32. Po trzecim członie następuje wymienienie numerów wszystkich rastrów sprawdzonych w danym kroku. Na przykład paczka oznaczająca sprawdzenie rastrów nr 105-110, oraz 115-118 przez robota nr 0 w pomieszczeniu P7 jest przedstawione jako: <**0.P7.32.105.106.107.108.109.110.115.116.117.118**>.

Przekazywanie „żetonu" następnemu obiektowi. Do informacji, że robot skończył już korzystać ze wspólnej wiedzy wykorzystano również protokół **kpo**. Pierwszy człon pozostaje bez zmian, drugi człon jest oznaczony jako -1 (nieaktywny), trzeci człon ma numer 11. W związku z tym rozszerzenie tablicy 3.3 uwzględniające wspólne sprawdzanie pomieszczenia, oraz przekazywanie „żetonu" następnemu robotowi, wygląda jak przedstawiono w tablicy 3.4. Na przykład przekazanie „żetonu" przez robota nr 2, następnemu robotowi (czyli robotowi nr 3, a jeżeli takiego nie ma to robotowi nr 0) wygląda następująco: <**2.-1.11**>.

Protokół kro

Protokół **kro** służy do wysyłania komunikatów z prośbą o przesunięcie się robota, który uniemożliwia przejazd nadawcy komunikatu. Drugi człon komunikatu określa robota, do

Nazwa operacji	*Numer operacji*
Sprawdzone rastry	32
Przekazanie „żetonu”	11

Tablica 3.4: Dodatkowe operacje przesyłane za pomocą protokołu **kpo** (rozszerzenie tablicy3.3)

.

którego skierowana jest prośba o przesunięcie się. Protokół ten ma tylko jedną operację oznaczoną numerem 50. Na przykład prośba wysłana przez robota nr 2 do robota nr 0 wygląda następująco: <**2.R0.50**>. Nie występuje odpowiedź na taką prośbę. Jeżeli adresat komunikatu nie może umożliwić przejazdu robotowi, który nadał komunikat, nadawca traktuje adresata jako zwykłą przeszkodę i omija ją.

3.4 Wiele robotów w budynku z poszukiwanymi obiektami

Zespół robotów z założenia ma znajdować obiekty w jak najkrótszym czasie. W związku z tym istota niniejszego modelu polega na optymalnym zachowaniu się robotów w momencie znalezienia poszukiwanego obiektu. Grupa zachowuje się następująco:

- Robot, który znajdzie poszukiwany obiekt wysyła komunikat z informacją o:
  - znalezionym obiekcie;
  - pomieszczeniu, w którym znalazł obiekt;
  - estymowanym kosztem sprawdzania pozostałej części pomieszczenia;
- Pozostałe roboty, w zależności od odległości do znalezionego obiektu i bieżącego trybu pracy, podejmują decyzję, czy są zainteresowane przerwaniem swojej bieżącej czynności i dojechaniem do pomieszczenia ze znalezionym obiektem. Jeżeli tak, przystępują do aukcji;
- Aukcja jest mechanizmem wyłaniającym roboty, które przerywają swoją bieżącą czynność i jadą do pomieszczenia ze znalezionym obiektem;
- Roboty, które wygrały aukcję muszą zwolnić rezerwowane przez siebie pomieszczenia, umożliwiając sprawdzenie ich przez inne roboty w późniejszym czasie;

W związku z takim schematem działania, niniejszy model musi być rozszerzony w stosunku do modelu opisanego w poprzednim podrozdziale o następujące elementy:

- estymowanie kosztu sprawdzania reszty pomieszczenia, przez robota, który znalazł poszukiwany obiekt;
- decyzyję robota, o zainteresowaniu aukcją;
- model aukcji;
- formaty komunikatów:
  - informujących o znalezieniu obiektu;
  - pozwalających uczestniczyć w aucji;
  - informujący o zwalnianiu zarezerwowanych pomieszczeń;
- sprawdzanie pomieszczeń, zwalnianych przez roboty, które wygrały aukcję;

3.4.1 Estymowanie kosztu sprawdzania reszty pomieszczenia

Z każdym poszukiwanym obiektem związana jest liczba określająca ile „kroków" robot musi poświęcić na obsłużenie obiektu. Może to oznaczać czas potrzebny na pomoc rannemu, lub czas potrzebny na rozbrojenie znalezionej bomby (dla każdego poszukiwanego obiektu ten czas może być różny). Robot po znalezieniu obiektu, wie ile „kroków" wykonał sprawdzając pomieszczenie, wie ile „kroków" wymaga obsługa obiektu, oraz wie jaką część pomieszczenia sprawdził. Na podstawie tych danych estymowany koszt sprawdzania całego pomieszczenia jest obliczany zgodnie ze wzorem 3.1.

$$k_{est} = \frac{P_c * k_{ob}}{P_r}, \tag{3.1}$$

gdzie:
k_{est} - estymowany koszt sprawdzania całego pomieszczenia,
P_c - ilość wszystkich rastrów w pomieszczeniu,
P_r - ilość sprawdzonych rastrów,
k_{ob} - ilość kroków wykonanych do momentu znalezienia i obsługi obiektu;

3.4.2 Aukcja

Decyzja o uczestnictwie w aukcji

Robot, który otrzyma informację o znalezieniu poszukiwanego obiektu oblicza koszt dojazdu do pomieszczenia, w którym znaleziono obiekt. Następnie znając estymowany koszt sprawdzania pomieszczenia, robot decyduje, czy jest zainteresowany przerwaniem bieżącej czynności i dojazdem do pomieszczenia, według wzoru 3.2

$$\frac{k_{est}}{u+1} > k_{doj} * \mu, \tag{3.2}$$

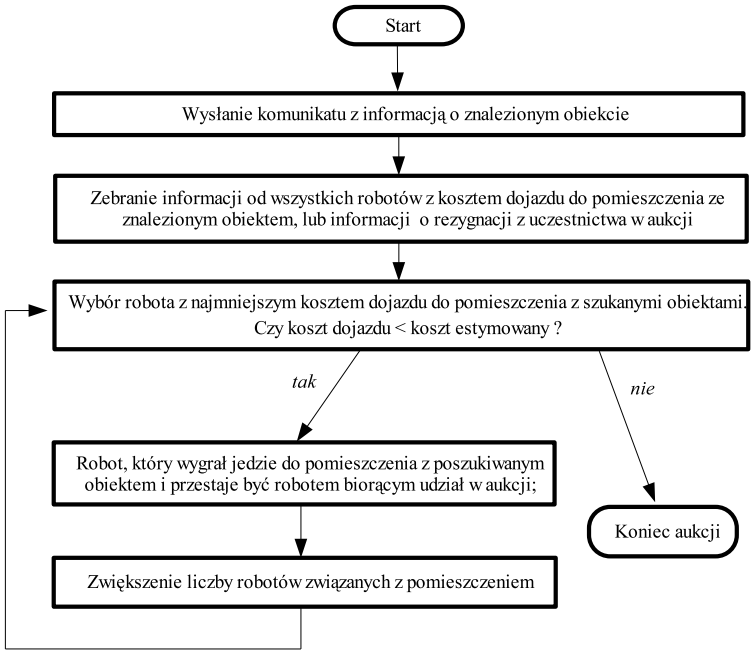
gdzie:
k_{doj} - koszt dojazdu do pomieszczenia ze znalezionym obiektem,
u - ilość robotów w pomieszczeniu, lub zmierzających do niego,
$\mu >= 1$ - współczynnik dojazdu, określający stosunek estymowanego kosztu sprawdzania pomieszczenia, do kosztu dojazdu. Standardowo $\mu = 1$. Rysunek 3.13 przedstawia sytuację, gdy współczynnik μ jest za mały.
Współczynnik dojazdu μ określa ile razy koszt sprawdzania pomieszczenia musi być większy niż koszt dojazdu, żeby robot był zainteresowany dojazdem. W toku badań eksperymentalnych współczynnik dojazdu μ będzie zmodyfikowany.
Jeżeli nierówność 3.2 zostanie spełniona, robot wysyła informację, że jest zainteresowany dojazdem do pomieszczenia. W przeciwnym przypadku, robot wysyła informację, że rezygnuje z udziału w aukcji.

Wygrywanie aukcji

Spośród wszystkich robotów zgłaszających się do aukcji wybierany jest robot, który znajduje się najbliżej. Ten robot wygrywa aukcję. Wszystkie pozostałe roboty ponownie sprawdzają nierówność 3.2, zwiększając wartość „u" (liczbę robotów będących lub jadących do pomieszczenia). Następnie ponownie wybierany jest robot z najmniejszym kosztem dojazdu i ten roboty również wygrywa aukcję. Pozostałe roboty ponownie zwiększają wartość „u" i sprawdzają nierówność 3.2, itd. Aukcja trwa do momentu, gdy robot z najmniejszym kosztem nie jest zainteresowany już dojazdem, gdyż nierówność 3.2 nie jest już spełniona. Mechanizm negocjacji jest przedstawiony na schemacie 3.16.

Postępowanie po wygraniu aukcji

Robot, który wygrał aukcję mógł znajdować się w jednym z dwóch trybów: albo dojeżdżał do pomieszczenia w celu sprawdzenia, albo sprawdzał pomieszczenie. Część pomieszczeń robot zarezerwował (czyli ustawił status: *zarezerwowane*), jedno pomieszczenie miało status, w zależności od bieżącego trybu robota, odpowiednio: *zarezerwowane-dojazd* lub *sprawdzane*. W przypadku gdy robot był w trybie dojazdu, wszystkie zarezerwowane pomieszczenia robot ustawia na *niesprawdzone*. W przypadku gdy robot sprawdził już część pomieszczenia, najpierw wysyła komunikat z informacją, które rastry w danym pomieszczeniu są sprawdzone a następnie wszystkie zarezerwowanie pomieszczenia oraz to, które sprawdza, ustawia na *niesprawdzone*. W ten sposób wszystkie roboty wiedzą jaka część pomieszczenia jest sprawdzona i robot, który będzie kontynuował sprawdzanie tego pomieszczenia, sprawdzi tylko niesprawdzoną jego część.
Roboty, które wygrają aukcję, udają się do pomieszczenia, w którym znaleziono poszukiwane obiekty i rozpoczynają sprawdzanie tego pomieszczenia. Mechanizm sprawdzania pomieszczenia przez kilka robotów został opisany w podrozdziale 3.3.3, przy okazji omawiania sytuacji, gdy któryś z robotów nie znajduje niesprawdzonego pomieszczenia i „pomaga" innemu robotowi sprawdzać jego pomieszczenie.



Rysunek 3.16: Schemat przedstawiający mechanizm aukcji robotów, wybierający roboty, które przerywają swoje bieżące działanie i jadą do pomieszczenia ze znalezionym obiektem.

3.4.3 Formaty komunikatów związanych z aukcją robotów

Niniejszy podrozdział jest rozszerzeniem komunikatów opisanych w podrozdziale 3.3.3. Komunikaty występujące w aukcji, informują o:

- znalezieniu poszukiwanego obiektu, wraz z numerem pomieszczenia oraz estymowanym kosztem sprawdzania całego pomieszczenia;
- uczestnictwie robota w aukcji, wraz z kosztem dojazdu, lub niezainteresowaniu aukcją;
- rezygnacji ze sprawdzania zarezerwowanych pomieszczeń, po wygraniu aukcji;

Wszystkie opisane komunikaty wykorzystują protokół **kpo**, opisany w podrozdziale 3.3.3.

Znalezienie poszukiwanego obiektu. Znalezienie poszukiwanego obiektu rozszerza protokół **kpo** o nazwę obiektu, którego dotyczy komunikacja, oraz listę sprawdzonych rastrów. Lista sprawdzonych rastrów jest potrzebna do wyliczenia sprawdzonej części pomieszczenia. Operacji znalezienia poszukiwanego obiektu została przypisana wartość liczbowa 30. Na przykład informacja od robota nr 0 o znalezieniu obiektu z identyfikatorem *b13* w pomieszczeniu P7, którego estymowany koszt sprawdzania reszty pomieszczenia wynosi 500 kroków, wygląda jak niżej:

<0.P7.30.b13.500.136.154.155.156.157.174.175.176.177.194.195.196.197>.
Wszystkie liczby za estymowanym kosztem (tutaj 500) oznaczają numery sprawdzonych rastrów.

Przystąpienie lub rezygnacja z aukcji Po otrzymaniu komunikatu z informacją o znalezieniu poszukiwanego obiektu, robot przystępuje lub rezygnuje z aukcji. Komunikat z informacją o przystąpieniu do aukcji posiada wartość liczbową 40. W przypadku, gdy robot rezygnuje z uczestnictwa w aukcji wysyła komunikat z kosztem dojazdu równym ∞. Na przykład wysłanie komunikatu przez robota numer 2, z informacją o przystąpieniu do aukcji w sprawie obiektu oznaczonego jako *b13*, z kosztem dojazdu do pomieszczenia P7, wynoszącym 15, wygląda następująco: **<2.P7.40.b13.15>**. Jest to odpowiedź na informację o znalezionym obiekcie z poprzedniego przykładu. Rezygnacja z aukcji w sprawie obiektu *b13*, przez robota nr 1 wygląda: **<1.P7.40.b13.99999>**.

Zwalnianie zarezerwowanych pomieszczeń. Po wygraniu aukcji robot, który był w trakcie sprawdzania wysyła komunikat z informacją o sprawdzonych rastrach. Komunikat ten jest został opisany w podrozdziale 3.3.3, przy okazji omawiania sprawdzania pomieszczenia przez kilka robotów. Wszystkie pomieszczenia, które robot zarezerwował lub sprawdza, są zwalniane, czyli oznaczane jako niesprawdzone. Format komunikatu zwalniającego pomieszczenie wynika z tablicy 3.3. Na przykład zwolnienie pomieszczenia P1 przez robota nr 2 wygląda następująco: **<2.P1.20>**.
W tablicy 3.5 zestawiono wszystkie komunikaty występujące w systemie, łącząc ze sobą tablicę 3.3, tablicę 3.4, oraz dodając komunikaty związane z mechanizmem aukcji.

Nazwa operacji	*Numer operacji*
Przekazanie „żetonu”	11
Status pomieszczenia: *Niesprawdzone*	20
Status pomieszczenia: *Zarezerwowane*	21
Status pomieszczenia: *Zarezerwowane-dojazd*	22
Status pomieszczenia: *Sprawdzane*	23
Status pomieszczenia: *Sprawdzone*	24
Znalezienie poszukiwanego obiektu	30
Sprawdzone rastry	32
Zgłoszenie do aukcji	40
Prośba o przesunięcie się innego robota	50
Wykluczenie robota z zespołu z powodu awarii	60

Tablica 3.5: Wszystkie komunikaty występujące w systemie i odpowiadająca im wartość liczbowa

.

3.4.4 Awaria robota

Prezentowany system jest całkowicie rozproszony i nie posiada jednostki centralnej. Jak opisano w podrozdziale 3.3.2 roboty komunikują się ze sobą na zasadzie „przeskakującego żetonu". To znaczy, że robot mający „żeton" może korzystać ze wspólnej bazy wiedzy, oraz wysyłać komunikaty. Następnie robot przekazuje „żeton" robotowi z numerem o jeden większym. Roboty wiedzą ilu jest członków zespołu i robot o najwyższym numerze przekazuje „żeton" robotowi z numerem 0.
W systemie rozróżniane są dwa rodzaje awarii: mechaniczna i komunikacyjna. W przypadku awarii mechanicznej nie są podejmowane żadne specjalne działania. Robot cały czas jest w trybie pracy i do końca zadania pozostaje w tym samym stanie (np. dojazdu do pomieszczenia). Takie zachowanie wynika z faktu, że nie zbudowano mechanizmu, który pozwalałby robotowi stwierdzić, że występuje awaria mechaniczna (np. stanie w miejscu przez dłuższy czas mimo ciągłych komend „jedź do przodu"). W przypadku awarii komunikacyjnej robot mimo, że jest sprawny mechanicznie staje się bezwartościowym członkiem zespołu. Taki robot przerywa swoje działanie i zatrzymuje się. Po otrzymaniu „żetonu" i odczekaniu określonego czasu (*timeout*) następny robot sam sobie przydziela „żeton" i wysyła komunikat o wykluczeniu robota z zespołu. Na przykład komunikat wykluczający robota nr 3 przez robota nr 4 wygląda następująco: $<$**4.R3.60**$>$.
W obydwu przypadkach awarii, jeżeli uszkodzony robot miał zarezerwowane pomieszczenia zostaną one sprawdzone w końcowej części zadania, gdy nie będzie już niezarezerwowanych pomieszczeń (zgodnie ze schematem 3.14). Taki mechanizm powoduje, że zespół jest *odporny* i mimo awarii robota jest w stanie zakończyć wykonywanie zadania.
Analiza zachowania robota w przypadku awarii nie była szczegółowo badana. Istotą niniejszego podrozdziału jest pokazanie, że prezentowany model jest w stanie kontynuować wykonywanie zadania mimo awarii robota. Ponieważ mimo iż robot z awarią komunikacji nie może się porozumiewać z resztą zespołu, może poruszać się i kontynuować poszukiwanie obiektów. Ciekawym rozszerzeniem pracy byłoby rozbudowanie zachowania robota o możliwość dalszej pracy robota w przypadku braku komunikacji z resztą zespołu.

Rozdział 4

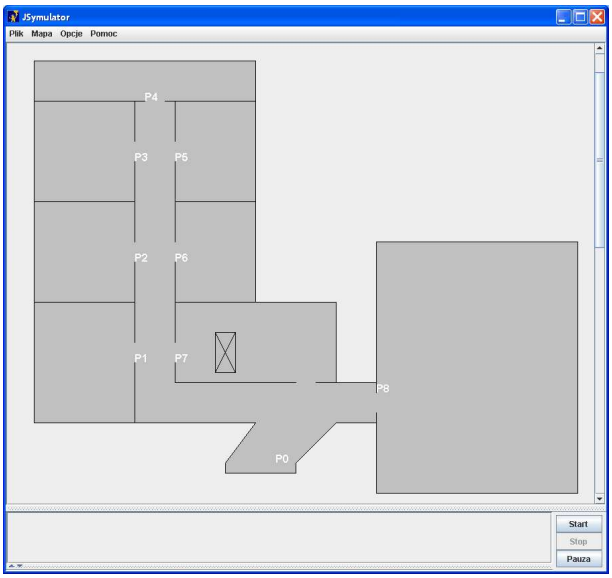
Eksperymenty

Szybkość z jaką poszukiwane obiekty zostaną znalezione zależy od:

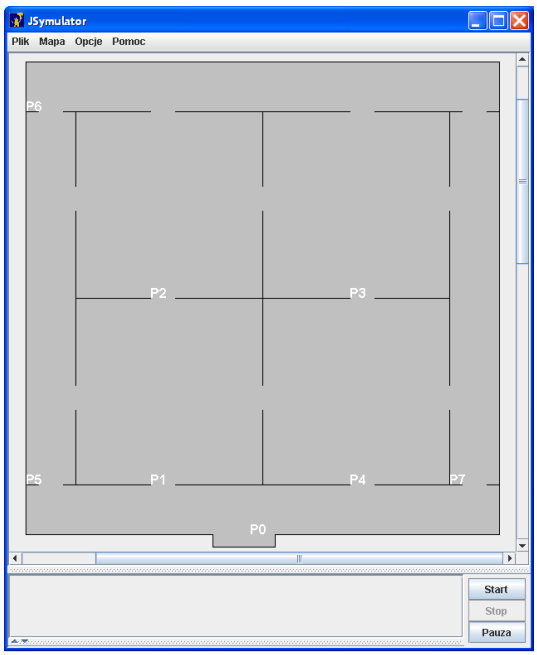
- mapy przeszukiwanego obszaru (ilości i rozkładu pomieszczeń);
- ilości poszukiwanych obiektów;
- rozmieszczenia obiektów (skupione blisko siebie, czy rozłożone w obszarze całej mapy);
- kosztu obsługi każdego obiektu;
- ilości robotów w zespole;
- pozycji początkowej każdego z robotów;
- współczynnika chęci współpracy μ z nierówności 3.2 określającego wolę robota do zmiany bieżącej akcji i udaniu się do pomieszczenia ze znalezionym obiektem;

Skuteczność zaproponowanego modelu opisnego w poprzednim rozdziale, zweryfikowano w badaniach symulacyjnych. Przeprowadzone badania polegały na zasymulowaniu pracy grupy robotów w następujących warunkach:

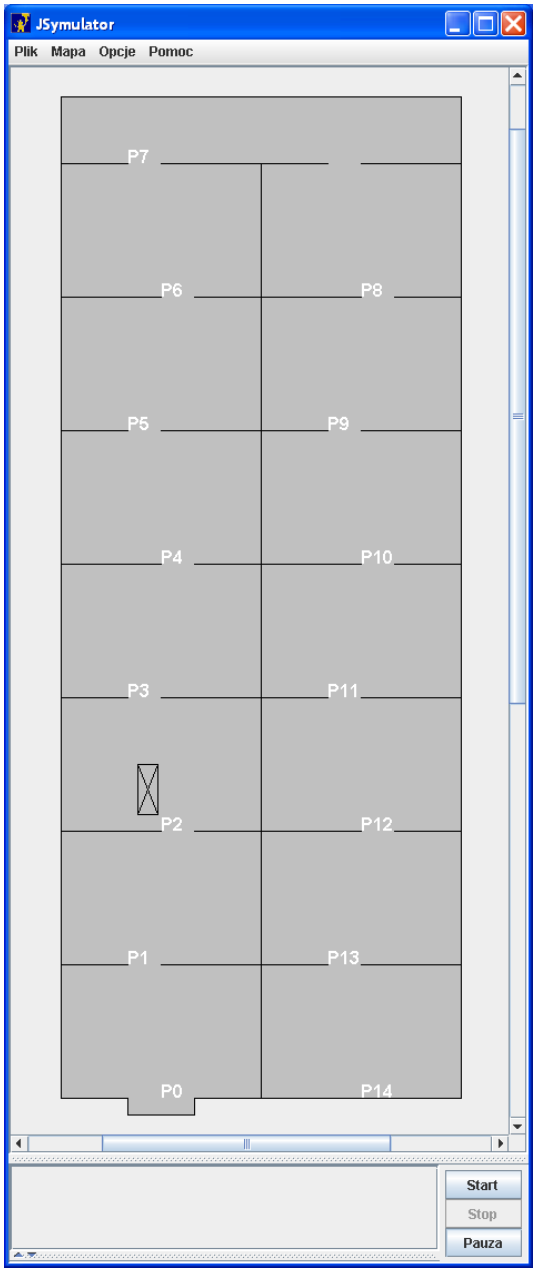
- trzy różne mapy, wszystkie ze stałym polem całego budynku (rysunki 4.1 - 4.3);
- równomierny i nierównomierny rozkład poszukiwanych obiektów w każdej mapie. Każda mapa zawiera taką samą liczbę poszukiwanych obiektów. Koszt obsługi jednego obiektu wynosi 500 kroków;
- różna ilość robotów (od 1 do 5);
- różne parametry współczynnika dojazdu μ przedstawionego w 3.2. Badano wartości $\mu = 1, 5, 10, 20, 50, \infty$;



Rysunek 4.1: Mapa typu *Szkoła*, w której jedno pomieszczenie jest znacznie większe od pozostałych. Może to przedstawiać np. szkołę z salą gimnastyczną, lub zwykły budynek biurowy;



Rysunek 4.2: Mapa typu *Każdy-z-każdym*, w której każde pomieszczenie jest połączone ze wszystkimi pomieszczeniami sąsiadującymi;



Rysunek 4.3: Mapa typu *Amfilada*, w której wszystkie pomieszczenia są umieszczone w jednej gałęzi grafu;

Dla każdego z powyższych przypadków przeprowadzono po 5 eksperymentów. Z każdego eksperymentu wyciągnięto wartości średnie, które wykorzystano do dalszej analizy. W eksperymantach przeprowadzonych z tymi samymi warunkami początkowymi otrzymano różne wyniki. Jest to zgodne z rzeczywistością, ponieważ nie można zakładać, że to samo doświadczenie wykonane przez jednego robota, lub kilka teoretycznie identy-

cznych robotów, da zawsze taki sam wynik.

W trakcie eksperymentów mierzono następujące parametry:

- średni koszt dojazdu do poszukiwanego obiektu;
- koszt dojazdu do ostatniego poszukiwanego obiektu;
- średnia ilość wygranych aukcji przypadających na jednego robota;

Dwa pierwsze parametry pokazują, że prezentowany system rzeczywiście działa. Zwiększenie liczby robotów w zespole, oraz zastosowanie zaawansowanego mechanizmu negocjacji istotnie zmniejszają średni oraz maksymalny koszt znalezienia poszukiwanego obiektu. Trzeci mierzony parametr - średnia ilość wygranych aukcji dla jednego robota jest potwierdzeniem prawidłowego działania systemu, to znaczy zwiększanie liczby robotów w zespole, lub zmniejszanie współczynnika chęci współpracy powoduje zwiększenie ilości wygranych aukcji przypadających na jednego robota.
Z powyższych założeń wynika, że ilość eksperymentów potrzebna do zrealizowania badań wynosi:
3(mapy) x 2(konfiguracje) x 5(zespołów robotów) x 6(współczynników współpracy) x 5(eksperymentów) = 900 eksperymentów.
Przeprowadzenie jednego eksperymentu trwa 20 - 30 minut, co daje około 14 dób ciągłej pracy komputera potrzebnej do przeprowadzenia całości badań.
W celu zwiększenia czytelności prezentowanej pracy w tabeli 4.1 zestawiono wszystkie rysunki i wykresy prezentowane w dalszej części niniejszego rozdziału.

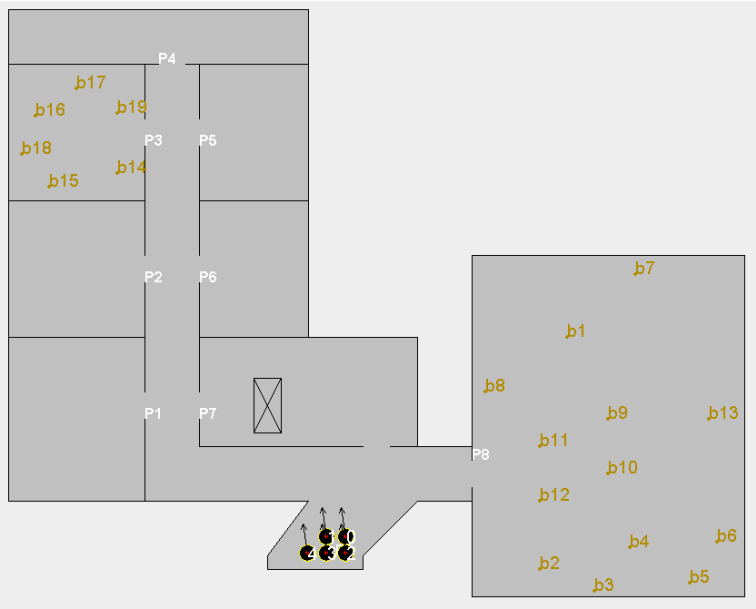
4.1 Mapa typu *Szkoła* ze skupionymi poszukiwanymi obiektami

Mapa typu *Szkoła* ze skupionymi poszukiwanymi obiektami jest przedstawiona na rysunku 4.4. Wszystkie poszukiwane obiekty są rozmieszczone w dwóch pomieszczeniach P3 i P8.

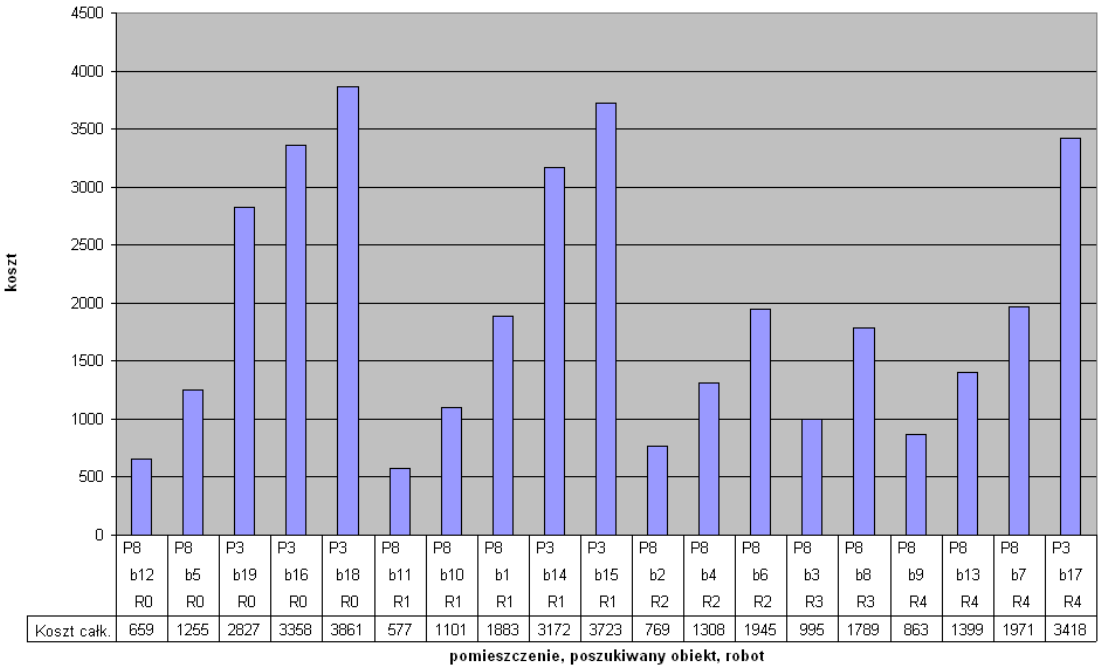
Wyniki jednego z eksperymentów zostały przedstawione na rysunku 4.5, z którego wynika, że dla danego przypadku, w wyniku zastosowanych negocjacji, wszystkie roboty sprawdzały pomieszczenie P8 i każdy z robotów znalazł po kilka obiektów. Roboty R0, R1, R2, R3 i R4 znalazły jako pierwsze obiekty odpowiednio: b12, b11, b2, b3 i b9. Koszt znalezienia każdego z obiektów w pomieszczeniu P8 był stosunkowo niski i wynosił 600 - 2000 kroków. Niestety kosztem szybkiego znalezienia obiektów w pomieszczeniu P8 było zdecydowanie dłuższe poszukiwanie obiektów w pomieszczeniu P3, których koszt wynosił 2800 - 3900 kroków. Mimo to średni koszt poszukiwania jednego obiektu jest mniejszy, niż w przypadku zespołu robotów, który nie wykorzystuje mechanizmu negocjacji $\mu = \infty$. Taki przypadek został przedstawiony na rysunku 4.6. Widać, że wszystkie obiekty znajdujące się w pomieszczeniu P8, zostały znalezione przez robota R1, natomiast obiekty znajdujące się w pomieszczeniu P3, zostały znalezione przez robota R2. W czasie,

			Eksperyment dla grupy 5 robotów		Koszt obiektu		
Typ mapy	Położenie obiektów	Mapa	μ=1	$\mu = \infty$	Średni	Maks.	Ilość wygranych aukcji
Szkoła	skupione	rys.4.4, str.66	rys.4.5, str.67	rys.4.6, str.69	rys.4.7, str.69	rys.4.8, str.70	rys.4.9, str.70
Szkoła	rozproszone	rys.4.10, str.71	rys.4.11, str.72	rys.4.12, str.73	rys.4.13, str.74	rys.4.14, str.74	rys.4.15, str.75
Amfilada	skupione	rys.4.16, str.77	rys.4.17, str.78	rys.4.18, str.79	rys.4.19, str.80	rys.4.20, str.80	rys.4.21, str.81
Amfilada	rozproszone	rys.4.22, str.82	rys.4.23, str.83	rys.4.24, str.83	rys.4.25, str.84	rys.4.26, str.84	rys.4.27, str.86
Każdy-z-każdym	skupione	rys.4.28, str.87	rys.4.29, str.88	rys.4.30, str.89	rys.4.31, str.90	rys.4.32, str.90	rys.4.33, str.91
Każdy-z-każdym	rozproszone	rys.4.34, str.92	rys.4.35, str.93	rys.4.36, str.93	rys.4.37, str.94	rys.4.38, str.94	rys.4.39, str.95

Tablica 4.1: Zestawienie rysunków prezentowanych w dalszej części niniejszego rozdziału;



Rysunek 4.4: Mapa typu *Szkoła*, ze skupionymi poszukiwanymi robotami. Na dole widoczny zespół 5 robotów znajdujący się w punkcie startu. Poszukiwane obiekty są rozmieszczone w pomieszczeniach P3 i P8;



	P8 b12 R0	P8 b5 R0	P3 b19 R0	P3 b16 R0	P3 b18 R0	P8 b11 R1	P8 b10 R1	P8 b1 R1	P3 b14 R1	P3 b15 R1	P8 b2 R2	P8 b4 R2	P8 b6 R2	P8 b3 R3	P8 b8 R3	P8 b9 R4	P8 b13 R4	P8 b7 R4	P3 b17 R4
Koszt całk.	659	1255	2827	3358	3861	577	1101	1883	3172	3723	769	1308	1945	995	1789	863	1399	1971	3418

Rysunek 4.5: Koszt całkowity znalezienia obiektów; mapa typu *Szkoła*, obiekty skupione, $\mu = 1$, grupa 5 robotów - jeden z przypadków

kiedy roboty R1 i R2 przeszukiwały pomieszczenia odpowienio P8 i P3, pozostałe roboty sprawdzały inne pomieszczenia, w których nie było poszukiwanych obiektów. Porównując rysunki 4.5 i 4.6, oraz uwzględniając założenie, że poszukiwane obiekty znajdują się w grupach, bardziej sensowne wydaje się zachowanie grupy wykorzystującej mechanizm aukcji i pomagającej sobie wzajemnie (μ małe), niż grupy nie negocjującej ze sobą ($\mu = \infty$).

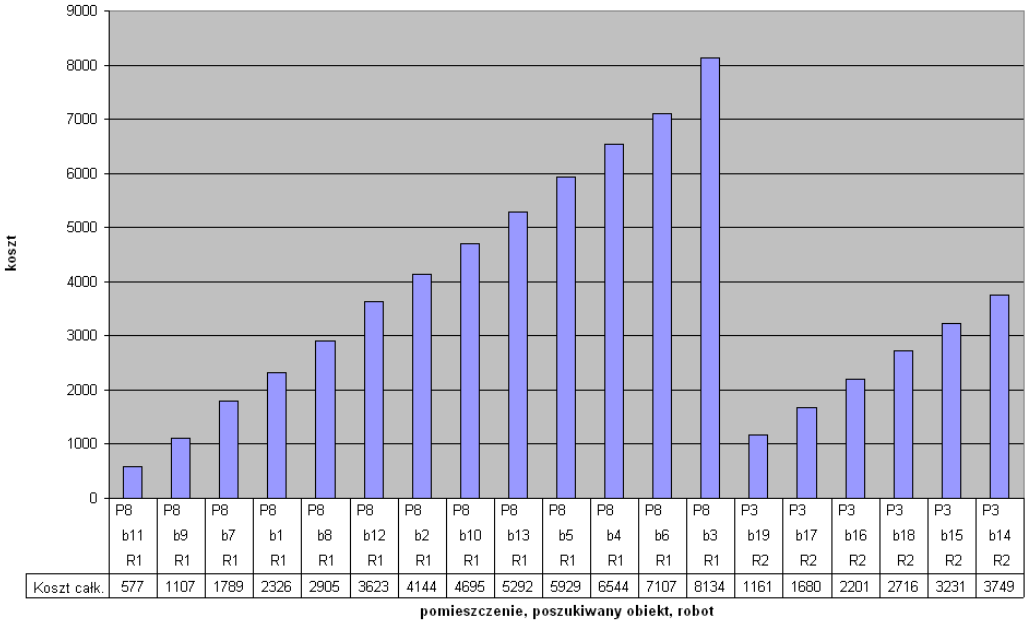
Na rysunku 4.7 zestawiono grupy składające się z 1 - 5 robotów, dla $\mu = 1$, 5, 10, 20, 50 i ∞. Rysunek przedstawiona wartości średnie z 5 identycznych eksperymentów. Ostatnia kolumna przedstawia wersję bez negocjacji ($\mu = \infty$). Z rysunku wyraźnie wynika, że zwiększenie liczby robotów przyspiesza wykonanie zadania. Dla jednego robota średni koszt dojazdu do pojedynczego obiektu wynosi 8366 kroków (średnia z pierwszej kolumny), podczas gdy dla grupy dwóch robotów, średni koszt dojazdu do jednego poszukiwanego obiektu wynosi już ponad dwa razy szybciej (średnio 3893 kroków - średnia z drugiej kolumny). Zwiększanie liczby robotów w grupie powoduje dalsze zmniejszanie kosztu dojazdu i dla grupy 5 robotów średni koszt dojazdu wynosi 2284 kroki (średnia z ostatniej kolumny).

To, że zwiększenie liczby robotów powoduje szybsze wykonanie zadania jest oczywiste. Ciekawy jest wpływ współczynnika chęci współpracy μ na szybkość znalezienia obiektów. W przypadku gdy nie ma współpracy, średni koszt znalezienia obiektu jest zawsze większy, niż przy istniejącej negocjacji (ostatni słupek jest zawsze większy od pozostałych dla

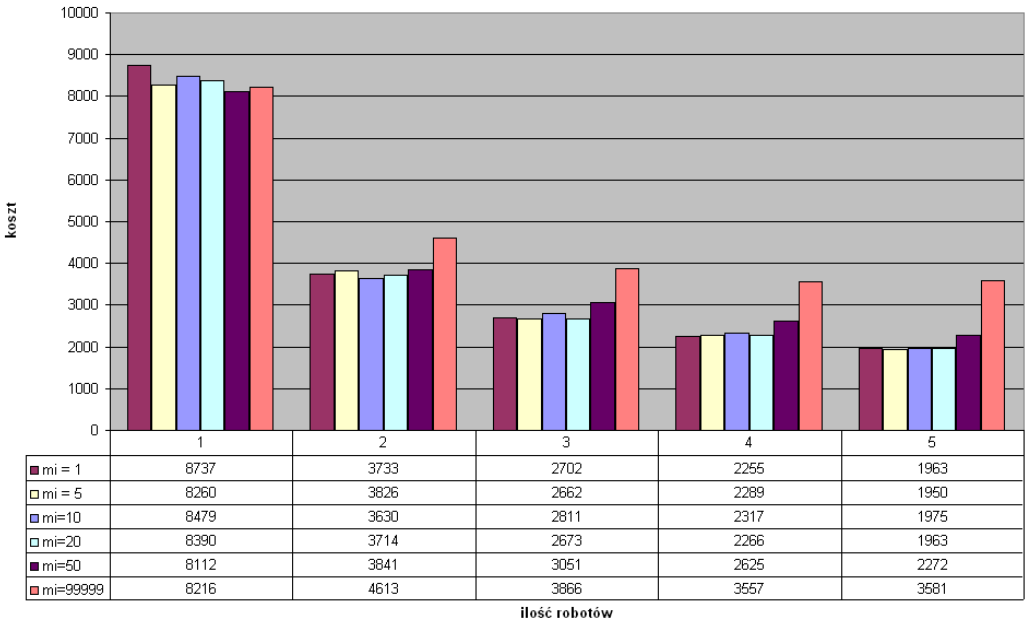
grupy składającej się więcej niż z jednego robota). W przypadku grupy składającej się z 5 robotów widać, że zastosowanie zaawansowanego mechanizmu aukcji powoduje zmniejszenie średniego kosztu znalezienia obiektu z 3581 kroków ($\mu = \infty$) do 1963 kroków (μ=1), czyli o 46%!

Z rysunku 4.7 wynika ponadto, że dla grupy 2 robotów, dobór współczynnika chęci dojazdu μ nie wpływa na średni koszt znalezienia obiektu (różnice dla różnych wartości μ są w granicach 5%). Dla grup składających się z więcej niż 2 robotów przy μ=50 (czyli mechanizm aukcji istnieje, ale roboty niechętnie z niego korzystają) wyniki są lepsze niż przy całkowitym braku negocjacji, ale gorsze niż przy większym wykorzystywaniu mechanizmu aukcji (mniejsze wartości μ). Największą różnicę widać w zespole składającym się z 4 robotów, gdzie dla μ<=20 średni koszt znalezienia obiektu wynosi 2282 kroki, a dla μ=50 koszt wynosi 2625 kroków, czyli około 15% więcej.
Rysunek 4.8 przedstawia ten sam przypadek, co rysunek 4.7, lecz zawiera koszt znalezienia ostatniego obiektu, zamiast średniego kosztu znalezienia obiektu. Porównując koszt znalezienia ostatniego obiektu grupie wykorzystującej i nie wykorzystującej mechanizmu aukcji, jeszcze wyraźniej widać zysk płynący z negocjacji robotów. Najwyraźniej widoczne jest to dla grupy 5 robotów (rys. 4.8), gdzie koszt znalezienia ostatniego obiektu, w grupie z $\mu = \infty$ wynosi 7992 kroki a dla μ=1 ostatni obiekt jest znaleziony po 4045 krokach, co przyspiesza znalezienie obiektu prawie o połowę!
Rysunek 4.9 przedstawia średnią ilość wygranych negocjacji przypadających na 1 robota we wszystkich rozpatrywanych przypadkach (różna ilość robotów w grupie i różny współczynnik chęci dojazdu). Ten wykres jest potwierdzeniem, że prezentowany system rzeczywiście działa. Najwięcej wygranych aukcji jest w przypadku największej liczby robotów i najmniejszego współczynnika chęci współpracy μ. Ze wzrostem współczynnika μ średnia ilość wygranych aukcji spada, ponieważ roboty są mniej spolegliwe. Również w przypadku zmniejszania liczby robotów w zespole spada średnia ilość wygranych aukcji, ponieważ roboty pracują w większych odległościach od siebie i trudniej jest spełnić nierówność 3.2.

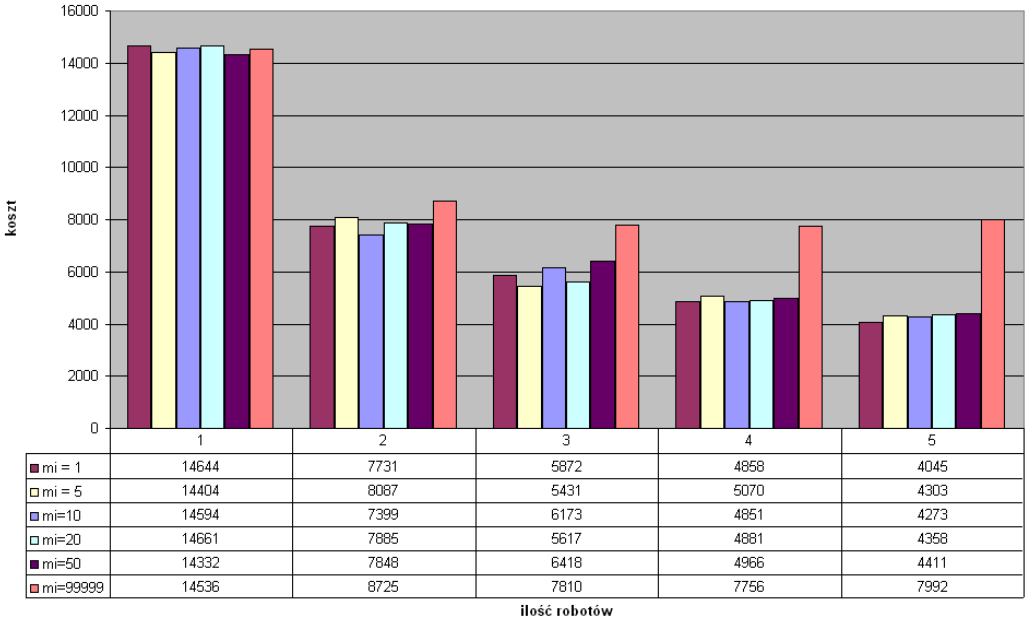
Średnia ilość wygranych negocjacji wynosi 0 w przypadku gdy $\mu = \infty$ (wtedy w ogóle nie ma negocjacji), oraz w przypadku gdy jest tylko jeden robot w grupie, co jest oczywiste. Na rysunku 4.9 widać, że dla μ=1 zwiększanie liczby robotów, powoduje jednoczesne zwiększenie ilości wygranych negocjacji. Wraz ze wzrostem μ widać pewną optymalną liczbę robotów, przekroczenie której powoduje zmniejszenie ilości wygranych aukcji. Dla μ=5 i μ=10, ilość wygranych aukcji jest porównywalna dla grupy 4 i 5 robotów, dla μ=20 i dla μ=50 największa średnia liczba wygranych aukcji jest dla grupy odpowiednio 4 i 5 robotów.



Rysunek 4.6: Koszt całkowity znalezienia obiektów; mapa typu *Szkoła*, obiekty skupione, $\mu = \infty$, grupa 5 robotów - jeden z przypadków

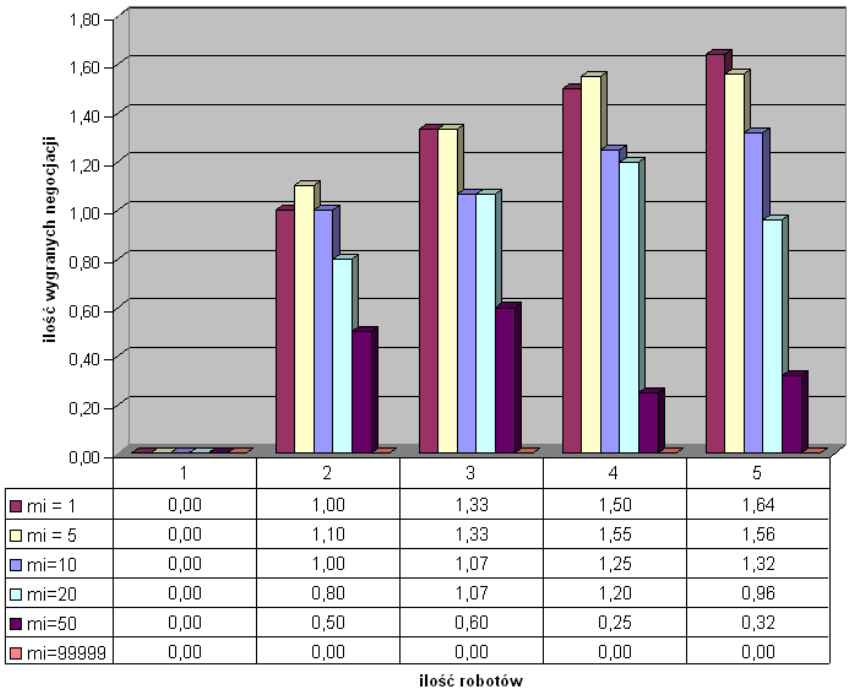


Rysunek 4.7: Średni koszt dojazdu do poszukiwanego obiektu w mapie typu *Szkoła*, obiekty skupione, dla różnej ilości robotów i różnego współczynnika μ;



	1	2	3	4	5
mi = 1	14644	7731	5872	4858	4045
mi = 5	14404	8087	5431	5070	4303
mi=10	14594	7399	6173	4851	4273
mi=20	14661	7885	5617	4881	4358
mi=50	14332	7848	6418	4966	4411
mi=99999	14536	8725	7810	7756	7992

Rysunek 4.8: koszt dojazdu do ostatniego poszukiwanego obiektu w mapie typu *Szkoła*, obiekty skupione, dla różnej ilości robotów i różnego współczynnika μ;

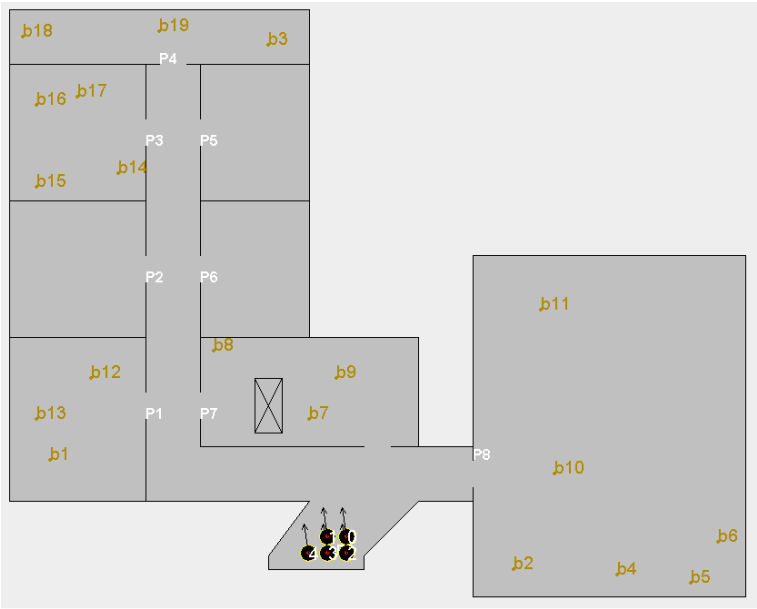


	1	2	3	4	5
mi = 1	0,00	1,00	1,33	1,50	1,64
mi = 5	0,00	1,10	1,33	1,55	1,56
mi=10	0,00	1,00	1,07	1,25	1,32
mi=20	0,00	0,80	1,07	1,20	0,96
mi=50	0,00	0,50	0,60	0,25	0,32
mi=99999	0,00	0,00	0,00	0,00	0,00

Rysunek 4.9: Średnia ilość wygranych negocjacji przypadających na jednego robota;

4.2 Mapa typu *Szkoła* z rozproszonymi poszukiwanymi obiektami

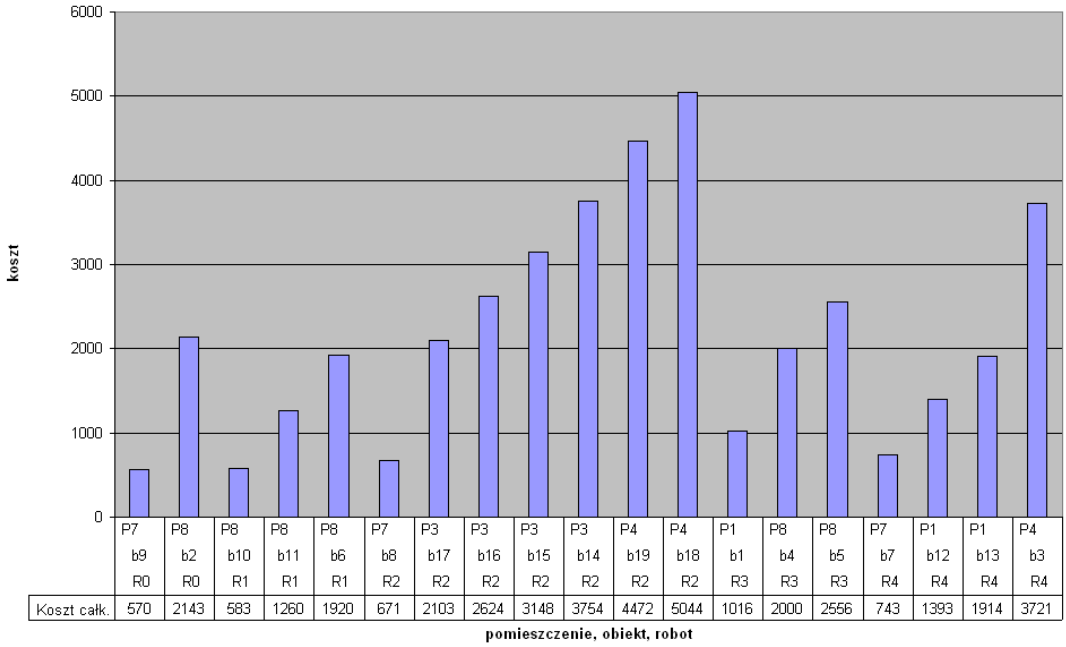
Mapa typu *Szkoła* z rozproszonymi poszukiwanymi obiektami jest przedstawiona na rysunku 4.10. Wszystkie poszukiwane obiekty są mniej więcej równomiernie rozmieszczone w obszarze całej mapy.



Rysunek 4.10: Mapa typu *Szkoła*, z rozproszonymi poszukiwanymi robotami. Na dole widoczny zespół 5 robotów znajdujący się w punkcie startu. Poszukiwane obiekty są rozmieszczone w miarę równomiernie w pomieszczeniach P1, P3, P4, P7 i P8;

Rysunki 4.11 i 4.12 przedstawiają po jednym z eksperymentów dla wartości μ=1, oraz $\mu = \infty$. Na rysunku 4.11 widać, że wszystkie obiekty zostały znalezione przez grupę 5 robotów w taki sposób, że grupa silnie wykorzystująca mechanizm negocjacji (μ=1), lepiej potrafiła podzielić między siebie poszukiwane obiekty, niż grupa nie negocjująca ze sobą ($\mu = \infty$) (tablica 4.2). Jednak częste wygrywanie aukcji i zmiana celu robota powoduje zwiększenie kosztu dojazdu do obiektu, co w efekcie powoduje nieznaczne pogarszenie skuteczności całego zespołu (koszt znalezienia ostatniego obiektu wynosi: 5000 kroków dla μ=1, oraz 4300 kroków dla $\mu = \infty$).

Na rysunku 4.13 zestawiono grupy składające się z 1 - 5 robotów, dla μ = 1, 5, 10, 20, 50 i ∞. Rysunek przedstawiona wartości średnie z 5 identycznych eksperymentów. Ostatnia kolumna przedstawia wersję bez negocjacji ($\mu = \infty$). Z rysunku wynika, że zwiększenie liczby robotów przyspiesza wykonanie zadania. Dla jednego robota średni koszt dojazdu do pojedynczego obiektu wynosi 7957 kroków, podczas gdy dla grupy dwóch robotów, koszt dojazdu do jednego poszukiwanego obiektu wynosi średnio 4181

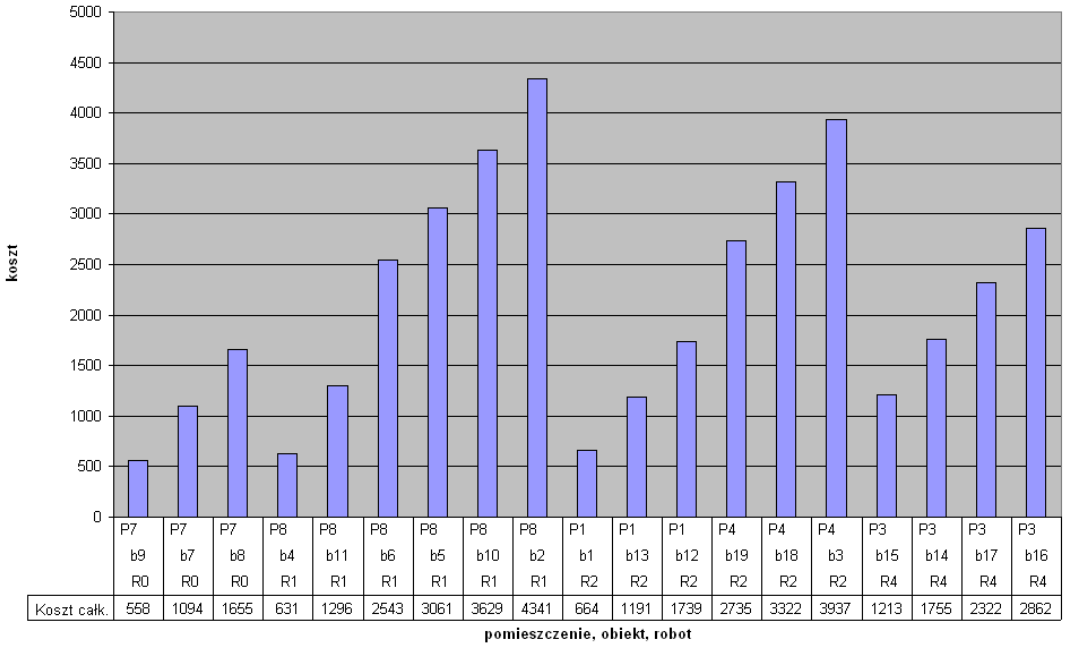


Rysunek 4.11: Koszt całkowity znalezienia obiektów; mapa typu *Szkoła*, obiekty rozproszone, $\mu = 1$, grupa 5 robotów - jeden z przypadków;

	Ilość znalezionych obiektów dla:	
Robot	μ=1	$\mu = \infty$
$R0$	2	3
$R1$	3	6
$R2$	7	6
$R3$	3	0
$R4$	4	4

Tablica 4.2: Numer robota i ilość znalezionych obiektów w grupie 5 robotów, dla μ=1, oraz dla $\mu = \infty$ - po jednym z przypadków;

kroków, czyli prawie dwa razy szybciej. Zwiększanie liczby robotów w grupie powoduje dalsze zmniejszanie kosztu dojazdu i dla grupy 5 robotów średni koszt dojazdu wynosi 2112 kroków. Zastosowanie zaawansowanego mechanizmu aukcji w grupie 1 i 2 robotów praktycznie nie wpływa na efektywność grupy (odchylenie średnie wynosi odpowiednio: 115 i 67 kroków, co stanowi około 1,5% wartości średniej). W grupie składającej się z 3, 4 lub 5 robotów zastosowanie zaawansowanego mechanizmu aukcji powoduje nieznaczny wzrost kosztu znalezienia obiektów, co jest spowodowane zmianą celu robota po wygranej aukcji.



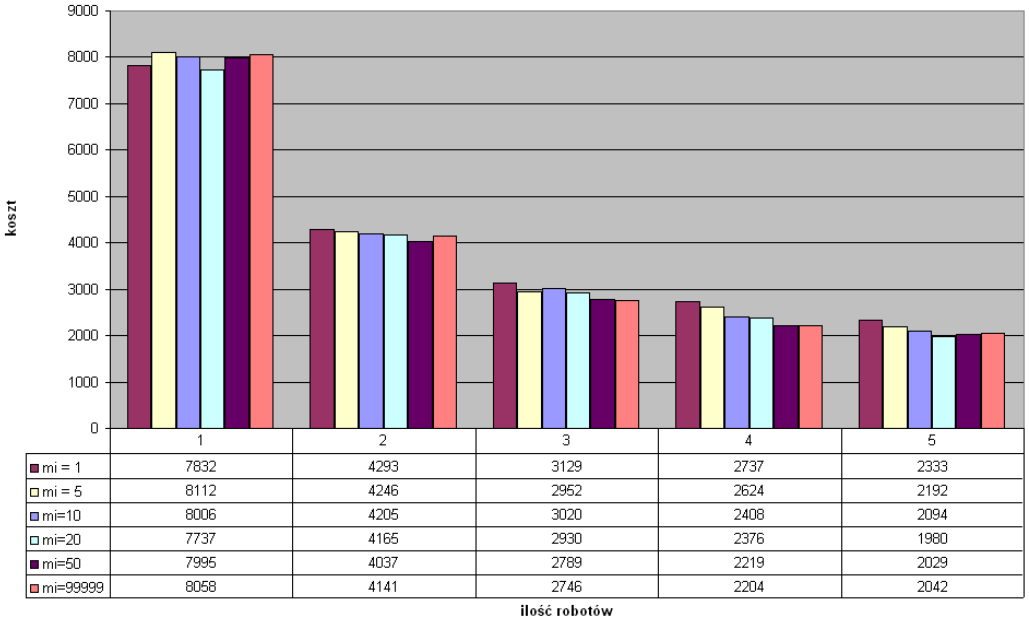
Rysunek 4.12: Koszt całkowity znalezienia obiektów; mapa typu *Szkoła*, obiekty rozproszone, $\mu = \infty$, grupa 5 robotów - jeden z przypadków;

Rysunek 4.15 przedstawia ilość średnią ilość wygranych negocjacji przypadających na 1 robota we wszystkich rozpatrywanych przypakach. Najwięcej wygranych aukcji jest w przypadku grupy 4 robotów, przy μ=1. Grupa 5 robotów ma nieznacznie mniejszą średnią wartość wygranych aukcji przypadających na 1 robota, ponieważ częściej występowały sytuacje, że ostatni robot przegrywał aukcję, ponieważ pozostałe 4 roboty wystarczały do sprawdzenia pomieszczenia. Z rysunku 4.15 widać, że wraz ze wzrostem współczynnika μ lub ze zmiejszaniem ilości robotów w zespole, maleje ilość wygranych aukcji przypadających na jednego robota, co potwierdza prawidłowe działanie prezentowanego modelu.

Porównanie poszukiwanych obiektów skupionych i rozproszonych w mapie typu *Szkoła*

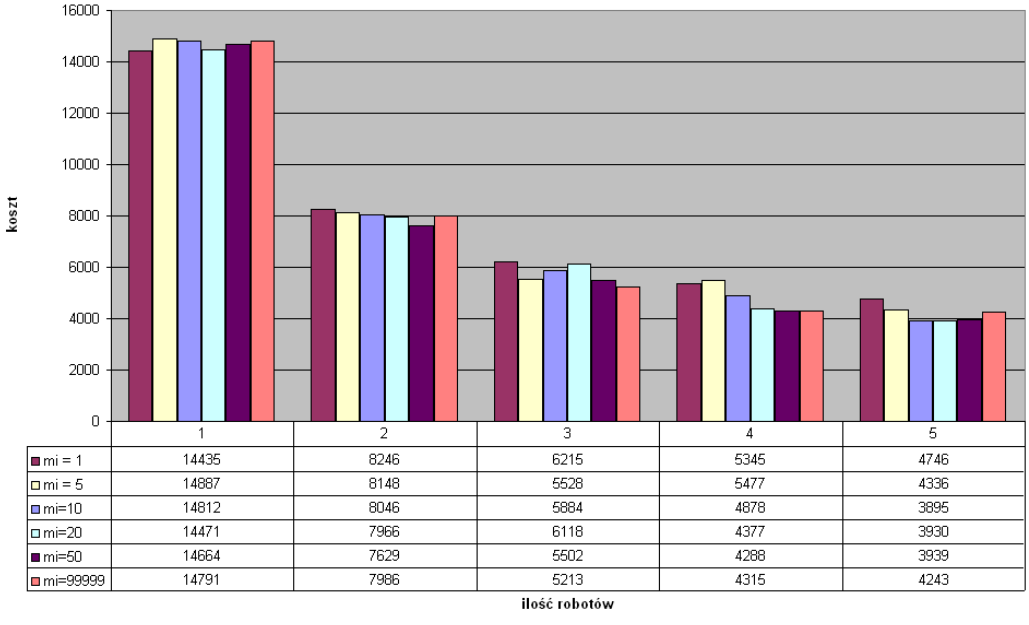
Porównując przypadki z tą samą mapą, z tą samą ilością robotów i tym samym współczynnikiem chęci współpracy μ a różniących się jedynie rozkładem poszukiwanych obiektów, można stwierdzić że:

- porównując rysunki 4.5, oraz 4.11 widać że dla μ=1 i grupy 5 robotów, koszt znalezienia ostatniego obiektu w przypadku obiektów skupionych i rozproszonych, wynosi odpowiednio około 4000 i 5000 kroków. Ostatni z obiektów skupionych został znaleziony szybciej, niż ostatni z obiektów rozproszonych. W przypadku



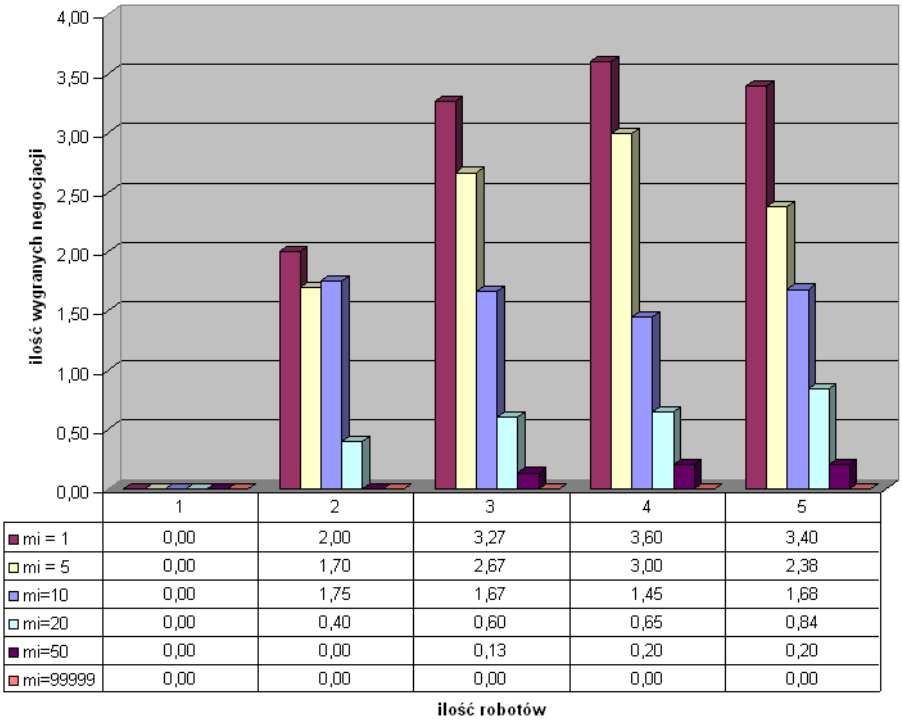
	1	2	3	4	5
mi = 1	7832	4293	3129	2737	2333
mi = 5	8112	4246	2952	2624	2192
mi=10	8006	4205	3020	2408	2094
mi=20	7737	4165	2930	2376	1980
mi=50	7995	4037	2789	2219	2029
mi=99999	8058	4141	2746	2204	2042

Rysunek 4.13: Średni koszt dojazdu do poszukiwanego obiektu w mapie typu *Szkoła*, obiekty rozproszone, dla różnej ilości robotów i różnego współczynnika μ



	1	2	3	4	5
mi = 1	14435	8246	6215	5345	4746
mi = 5	14887	8148	5528	5477	4336
mi=10	14812	8046	5884	4878	3895
mi=20	14471	7966	6118	4377	3930
mi=50	14664	7629	5502	4288	3939
mi=99999	14791	7986	5213	4315	4243

Rysunek 4.14: koszt dojazdu do ostatniego poszukiwanego obiektu w mapie typu *Szkoła*, obiekty rozproszone, dla różnej ilości robotów i różnego współczynnika μ;



	1	2	3	4	5
mi = 1	0,00	2,00	3,27	3,60	3,40
mi = 5	0,00	1,70	2,67	3,00	2,38
mi=10	0,00	1,75	1,67	1,45	1,68
mi=20	0,00	0,40	0,60	0,65	0,84
mi=50	0,00	0,00	0,13	0,20	0,20
mi=99999	0,00	0,00	0,00	0,00	0,00

Rysunek 4.15: Średnia ilość wygranych negocjacji przypadających na jednego robota w mapie typu *Szkoła*, obiekty rozproszone, dla różnej ilości robotów i różnego współczynnika μ;

obiektów rozproszonych większa jest szansa zdarzenia, że robot zaniecha jazdy do pomieszczenia, w którym jest poszukiwany obiekt i po wygraniu aukcji zmieni swój cel. Wtedy taki obiekt zostanie znaleziony później;

- porównując rysunki 4.6, oraz 4.12 widać, że dla $\mu = \infty$ i grupy 5 robotów, koszt znalezienia ostatniego obiektu w przypadku obiektów skupionych i rozproszonych, wynosi odpowiednio około 8000 i 4500 kroków. Zdecydowanie dłuższe poszukiwanie w przypadku obiektów skupionych wynika z faktu, że skoro nie istnieje mechanizm aukcji, czyli roboty sobie nawzajem nie pomagają, to cała grupa musiała czekać aż jeden robot znajdzie wszystkie obiekty w nawiększym pomieszczeniu;

- w przypadku równomiernego rozmieszczenia obiektów w obszarze całej mapy, obiekty są równocześnie znajdowane przez zespół robotów i mechanizm negocjacji jest niepotrzebny. W przypadku obiektów skupionych, można potraktować jedno pomieszczenie jako podobszar, w którym obiekty są równomiernie rozproszone i zastowanie mechanizmu negocjacji umożliwia „ściągnięcie" większej liczby robotów, w celu przeszukania danego podobszaru;

- porównując rysunki 4.9 i 4.15 widać, że w przypadku obiektów rozproszonych średnia

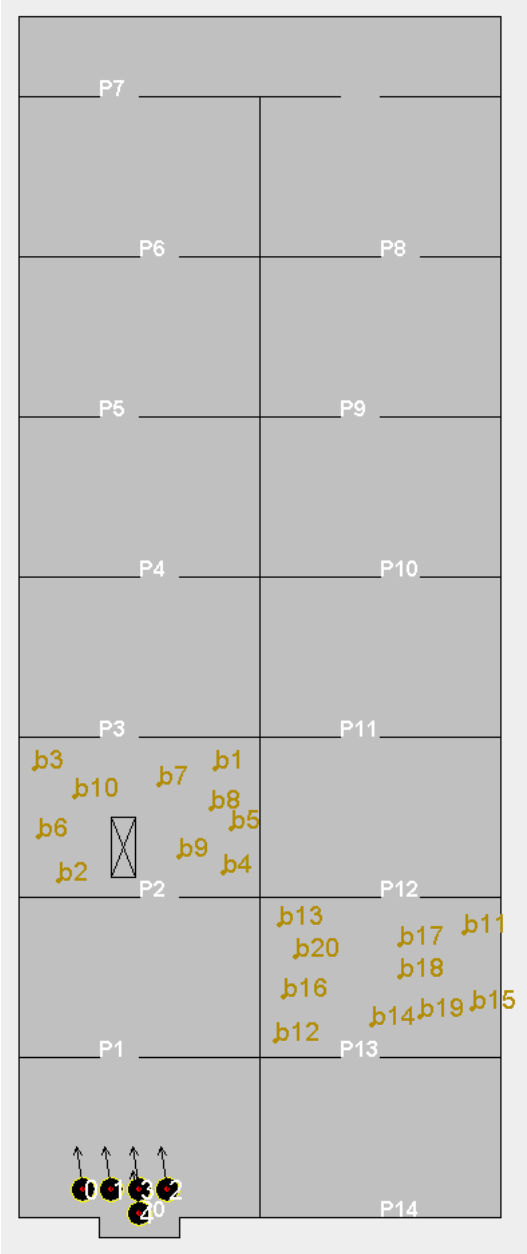
ilość wygranych aukcji jest znacznie większa niż w przypadku obiektów skupionych - maksymalna wartość wynosi odpowiednio: 3,60 i 1,64). Wynika to z faktu, że robot, który jest w pomieszczeniu, w którym znajdują się poszukiwane obiekty, lub jest w trakcie dojazdu do niego, nie jest zainteresowany innymi aukcjami. W przypadku obiektów skupionych taka sytuacja występuje znacznie częściej, niż w przypadku obiektów rozproszonych.

4.3 Mapa typu *Amfilada* ze skupionymi poszukiwanymi obiektami

Mapa typu *Amfilada* ze skupionymi poszukiwanymi obiektami jest przedstawiona na rysunku 4.16. Mapa charakteryzuje się szeregowym ułożeniem pomieszczeń - „jedno za drugim". Graf tej mapy zawierający w węzłach pomieszczenia lub drzwi pomiędzy pomieszczeniami, ma tylko jedną gałąź. Można przyjąć że dla prezentowanego systemu jest to natrudniejszy rodzaj mapy, ponieważ grupa robotów nie może się rozdzielić na inne gałęzie, musi wspólnie eksplorować tę samą gałąź grafu i będzie sobie wzajemnie przeszkadzać.

Poszukiwane obiekty są rozmieszczone w pomieszczeniach P2 i P13. Rysunki 4.17 i 4.18 przedstawiają po jednym z przeprowadzonych eksperymentów dla skrajnych wartości μ. Widać z nich, że w przypadku μ=1, czyli maksymalnego wykorzystywania mechanizmu aukcji, obiekty są znalezione przez wszystkie 5 robotów. W przypadku, gdy $\mu = \infty$ obiekty są znalezione tylko przez trzy roboty. Robot R0 jako jedyny sprawdził pomieszczenie P13, natomiast pomieszczenie P2 zostało sprawdzone przez 2 roboty (R1 i R4), co bardzo wyraźnie zmniejszyło koszt znalezienia obiektów b7, b10, b6 i b2 (czyli wszystkich znalezionych przez robota R4). Stało się tak dlatego, że robot R0 zarezerwował całą gałąź i zaczął ją sprawdzać od końca, czyli od pomieszczenia P14. Pozostałe roboty nie mogąc zarezerwować swojej gałęzi kolejno sprawdzały najbliższe niesprawdzone pomieszczenia. W przypadku wykorzystującym mechanizm aukcji ($\mu = 1$) roboty pomagały sobie wzajemnie, razem sprawdziły najpierw pomieszczenie P2 a następnie pomieszczenie P13. Koszt znalezienia ostatniego obiektu w przypadku μ=1 wynosi poniżej 4500 kroków a dla $\mu = \infty$ ponad 6000 kroków.
Rysunki 4.19 i 4.20 przedstawiają średni i maksymalny koszt dojazdu do obiektu, dla grup składających się od 1 do 5 robotów, dla różnych wartości współczynnika chęci współpracy μ. Z rysunków tych wynika, że:

- dla grupy 2 robotów, nadmierna chęć robotów do współpracy (małe μ) powoduje nieznaczne spowolnienie znalezienia obiektów. Koszt znalezienia ostatniego obiektu dla grupy 2 robotów, oraz μ=1 i μ=5 wynosi odpowiednio 7114 i 6639 kroków, wobec poniżej 6200 kroków w pozostałych przypadkach;
- dla grupy składającej się z 4 i 5 robotów mała wartość współczynnika μ poprawia



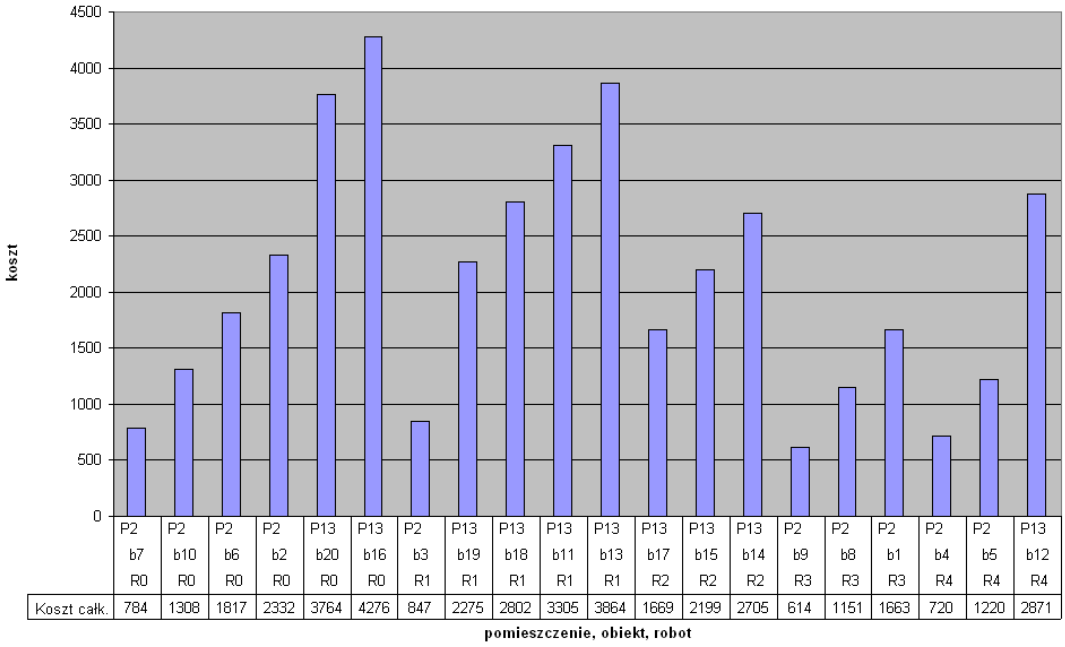
Rysunek 4.16: Mapa typu *Amfilada*, ze skupionymi poszukiwanymi robotami charakteryzująca się umieszczeniem pomieszczeń w stylu „jedno za drugim”;

działanie systemu. Koszt znalezienia ostatniego obiektu dla grupy 4 i 5 robotów i μ=1 wynosi około 4700 kroków, wobec około 6000 kroków w pozostałych przypadkach;

- dla wysokich wartości μ zwiększanie liczby robotów powyżej 2, nie przynosi żadnej

zmiany w skuteczności grupy. Wynika to z faktu, że jeden robot pracuje w jednym końcu budynku a pozostałe roboty pracują razem w drugim końcu. Żeby dojechać do zarezerwowanego pomieszczenia robot musi przejechać przez pomieszczenie sprawdzane przez innego robota. Zysk spowodowany zwiększeniem ilości robotów, jest tracony w „tłoku", w którym roboty pracują;

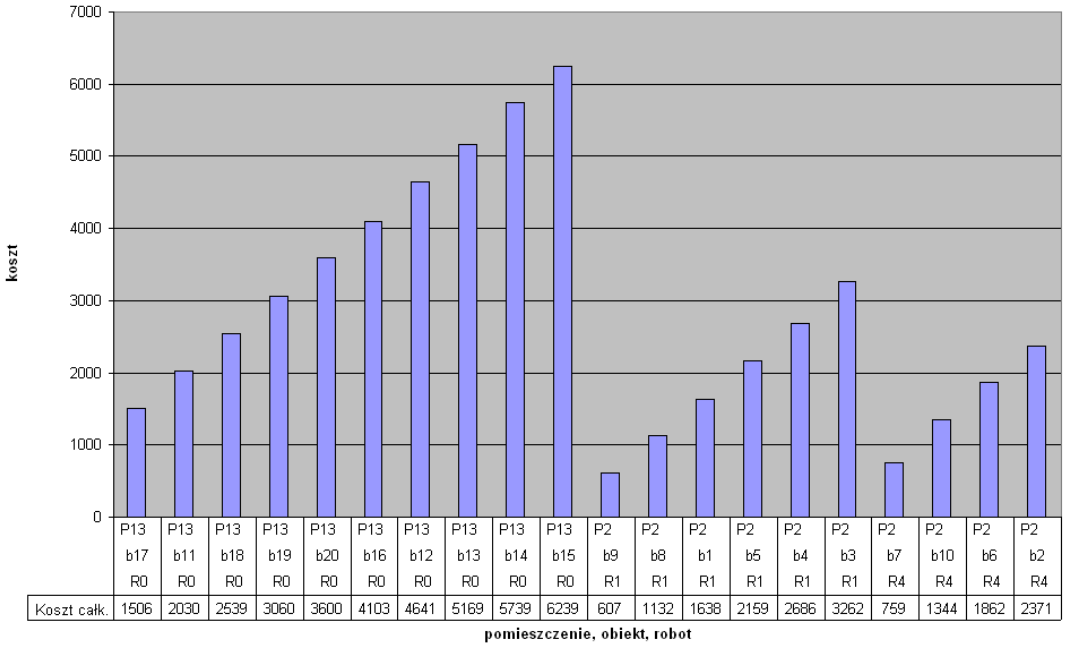
- dla małych wartości μ zwiększanie liczby robotów w zespole poprawia efektywność całej grupy. Jest to spowodowane tym, że w przeciwieństwie do poprzedniego podpunktu, roboty bardziej się rozproszą i zamiast pracy w trybie „jeden robot na jednym końcu - reszta na drugim", roboty podzielą się bardziej równomiernie;
- dla rozpatrywanej mapy najlepiej zastosować grupę 2 robotów, bez mechanizmu negocjacji, lub grupę 4 robotów wykorzystującą ten mechanizm;



	P2 b7 R0	P2 b10 R0	P2 b6 R0	P2 b2 R0	P13 b20 R0	P13 b16 R0	P2 b3 R1	P13 b19 R1	P13 b18 R1	P13 b11 R1	P13 b13 R1	P13 b17 R2	P13 b15 R2	P13 b14 R2	P2 b9 R3	P2 b8 R3	P2 b1 R3	P2 b4 R4	P2 b5 R4	P13 b12 R4
Koszt całk.	784	1308	1817	2332	3764	4276	847	2275	2802	3305	3864	1669	2199	2705	614	1151	1663	720	1220	2871

Rysunek 4.17: Koszt całkowity znalezienia obiektów; mapa typu *Amfilada*, obiekty skupione, μ=1, grupa 5 robotów - jeden z przypadków;

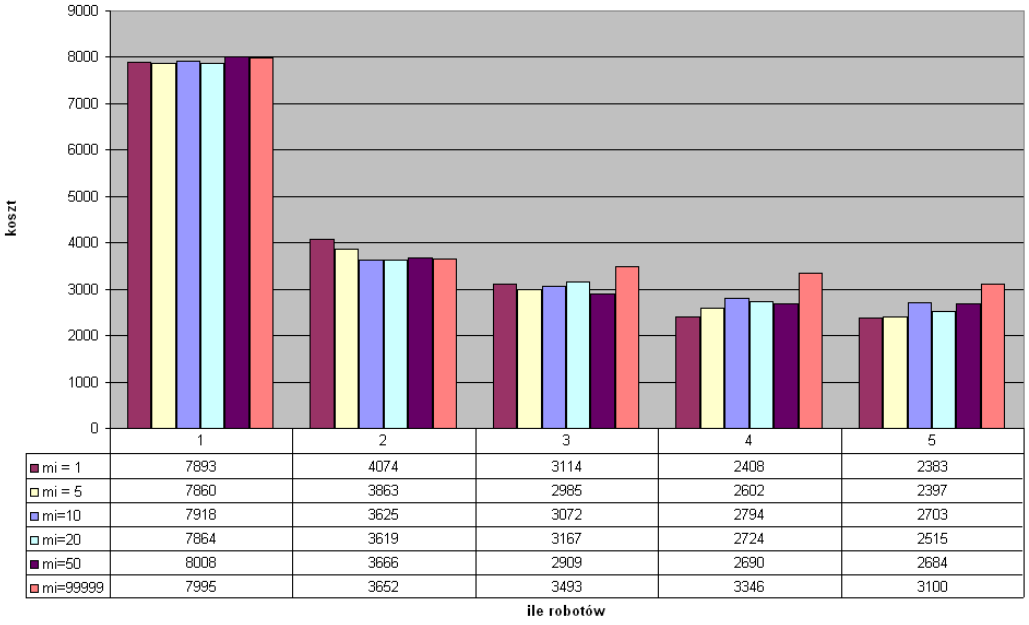
Wyżej opisane właściwości wynikają z tego, że rozpatrywana mapa charakteryzuje się bardzo dużą odległością między skrajnymi pomieszczeniami. Roboty, które wygrają aukcję mogą mieć do pokonania długą i kosztowną drogę do pomieszczenia, w którym są poszukiwane obiekty (dlatego mały współczynnik μ w grupe z małą ilością robotów nie sprawdza się). Jeżeli mamy do dyspozycji więcej robotów w zespole, wtedy roboty naturalnie rozproszą się w większym obszarze i aukcje będą wygrywały roboty najbliżej pomieszczenia z poszukiwanymi obiektami. Wtedy roboty znajdujące się w drugim końcu



Rysunek 4.18: Koszt całkowity znalezienia obiektów; mapa typu *Amfilada*, obiekty skupione, $\mu = \infty$, grupa 5 robotów - jeden z przypadków;

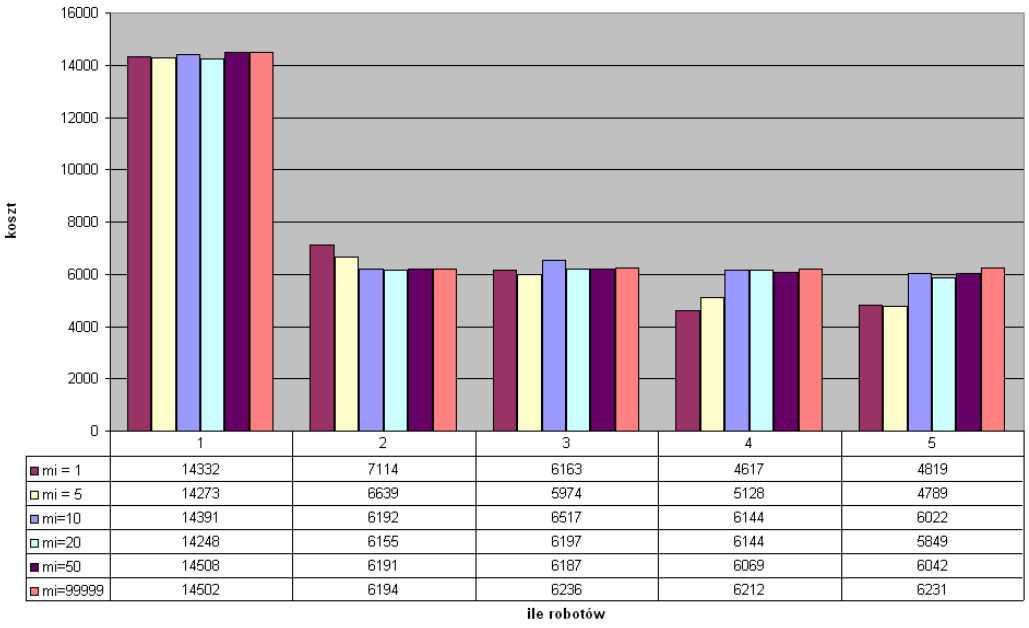
mapy nie będą zainteresowane wygraniem aukcji. Należy się spodziewać, że zbyt duża liczba robotów w efekcie zmniejszy skuteczność zespołu, z powodu jedoczesnego przejazdu tą samą ścieżką (na przykład przejazd przez drzwi).

Na rysunku 4.21 widać, że zespoły z małym współczynnikiem chęci współpracy ($\mu \leq 5$) wygrały zdecydowanie więcej aukcji, niż pozostałe. Zwłaszcza dla grupy 4 i 5 robotów i μ=1 lub μ=5, średnia liczba wygranych aukcji jest co najmniej 1.2. Potwierdza to prezentowane wyżej wnioski o największej skuteczności takich zespołów. Dla grupy 3 robotów i μ=1, średnia ilość wygranych aukcji nie wpływa na skuteczność grupy, podczas gdy stosunkowo wysoka ilość wygranych aukcji w przypadku zespołu dwurobotowego nieznacznie pogarsza jego działanie.



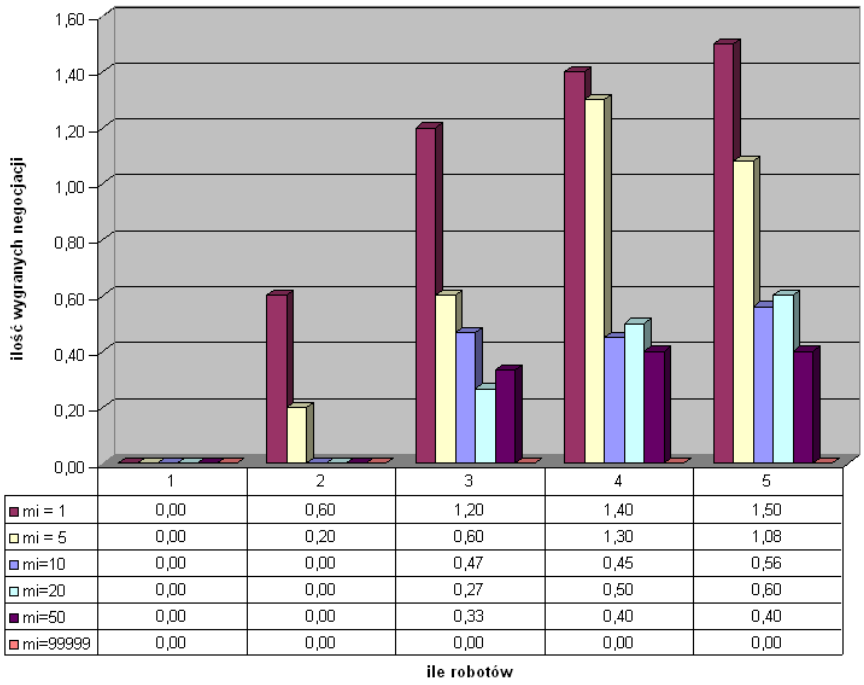
	1	2	3	4	5
mi = 1	7893	4074	3114	2408	2383
mi = 5	7860	3863	2985	2602	2397
mi=10	7918	3625	3072	2794	2703
mi=20	7864	3619	3167	2724	2515
mi=50	8008	3666	2909	2690	2684
mi=99999	7995	3652	3493	3346	3100

Rysunek 4.19: Średni koszt dojazdu do poszukiwanego obiektu w mapie typu *Amfilada*, obiekty skupione, dla różnej ilości robotów i różnego współczynnika μ



	1	2	3	4	5
mi = 1	14332	7114	6163	4617	4819
mi = 5	14273	6639	5974	5128	4789
mi=10	14391	6192	6517	6144	6022
mi=20	14248	6155	6197	6144	5849
mi=50	14508	6191	6187	6069	6042
mi=99999	14502	6194	6236	6212	6231

Rysunek 4.20: Koszt dojazdu do ostatniego poszukiwanego obiektu w mapie typu *Amfilada*, obiekty skupione, dla różnej ilości robotów i różnego współczynnika μ;



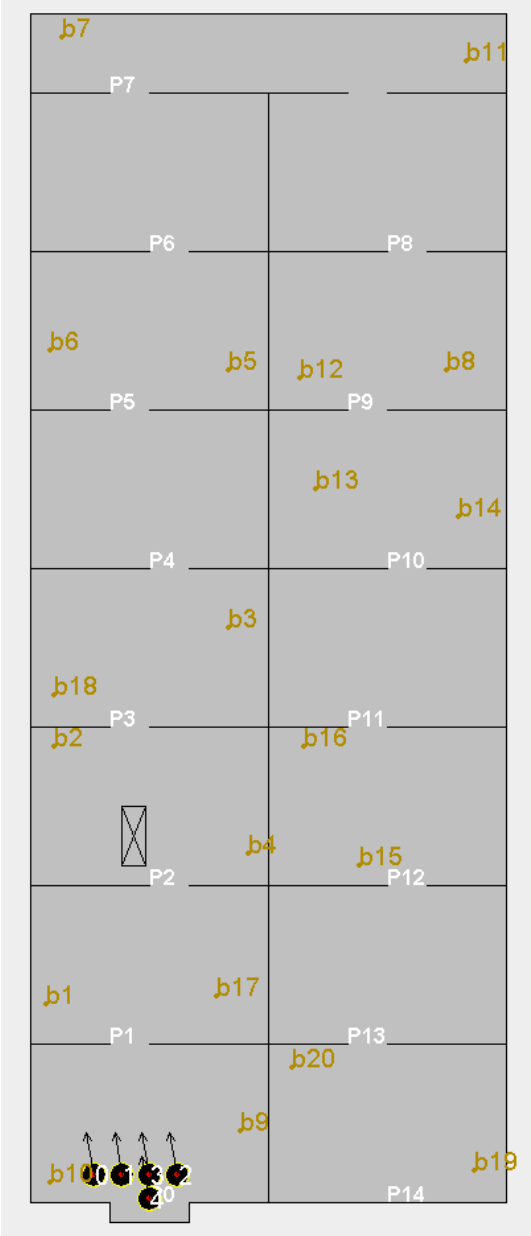
	1	2	3	4	5
mi = 1	0,00	0,60	1,20	1,40	1,50
mi = 5	0,00	0,20	0,60	1,30	1,08
mi=10	0,00	0,00	0,47	0,45	0,56
mi=20	0,00	0,00	0,27	0,50	0,60
mi=50	0,00	0,00	0,33	0,40	0,40
mi=99999	0,00	0,00	0,00	0,00	0,00

Rysunek 4.21: Średnia ilość wygranych negocjacji przypadających na jednego robota w mapie typu *Amfilada*, obiekty skupione, dla różnej ilości robotów i różnego współczynnika μ;

4.4 Mapa typu *Amfilada* z rozproszonymi poszukiwanymi obiektami

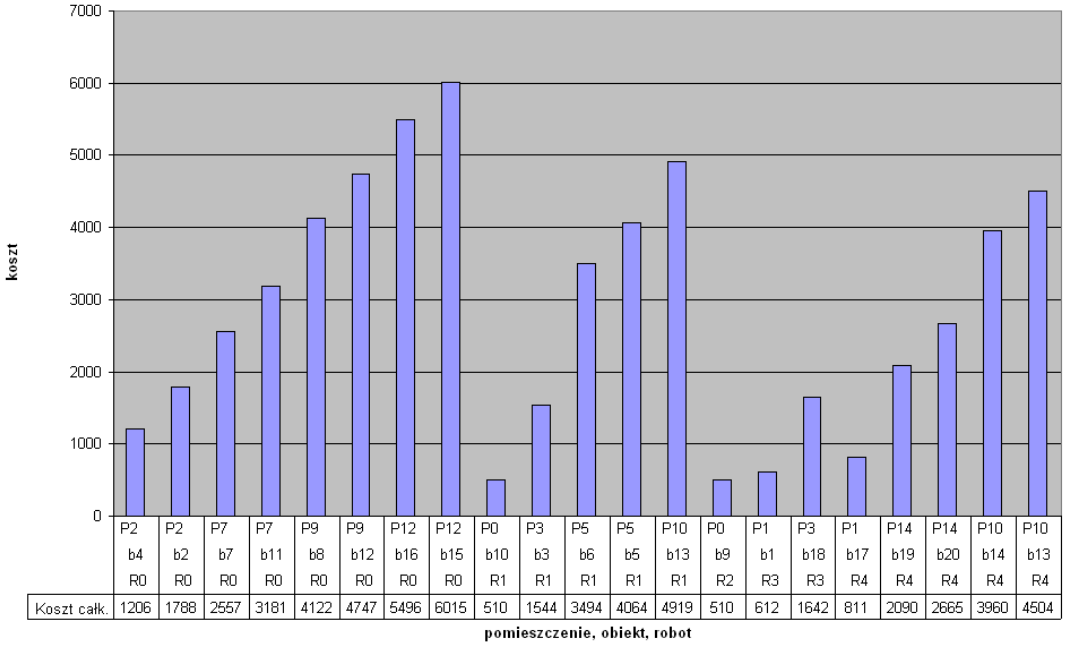
Mapa typu *Amfilada* z rozproszonymi poszukiwanymi obiektami jest przedstawiona na rysunku 4.22. Charakteryzuje się ona tym, że obiekty są rozrzucone w obszarze całej mapy. Po jednym z przeprowadzonych eksperymentów dla μ=1 i $\mu = \infty$, dla grupy 5 robotów przedstawiają rysunki 4.23 i 4.24. Ponieważ opisywana mapa jest najtrudniejszym z rozpatrywanych przypadków, poniżej szczegółowo opisano zachowanie grupy.

Dla μ=1, czyli maksymalnego wykorzystania mechanizmu negocjacji, w tym konkretnym eksperymencie, robot R4 jako pierwszy zarezerwował całą gałąź i rozpoczął dojazd do pomieszczenia P14. Ponieważ w chwili początkowej robot R4 znajdował się z tyłu grupy (rys. 4.22), wyjazd z pomieszczenia P0 dla robota R4 był utrudniony. Pozostałe roboty nie mogąc zarezerwować dla siebie innych gałęzi, rozpoczęły sprawdzanie najbliższych wolnych pomieszczeń. I tak: roboty R1 i R2 rozpoczęły przeszukiwanie pomieszczenia P0, a robot R3 rozpoczął przeszukiwanie pomieszczenia P1, po czym znalazł obiekt b1 i rozpoczął aukcję, którą wygrał robot R4, będący w trakcie dojazdu do pomieszczenia P4. Pozostałe roboty nie były zainteresowane aukcją, ponieważ znajdowały się w pomieszczeniu, w którym już znalazły poszukiwane obiekty. Roboty R3 i R4 wspólnie sprawdziły



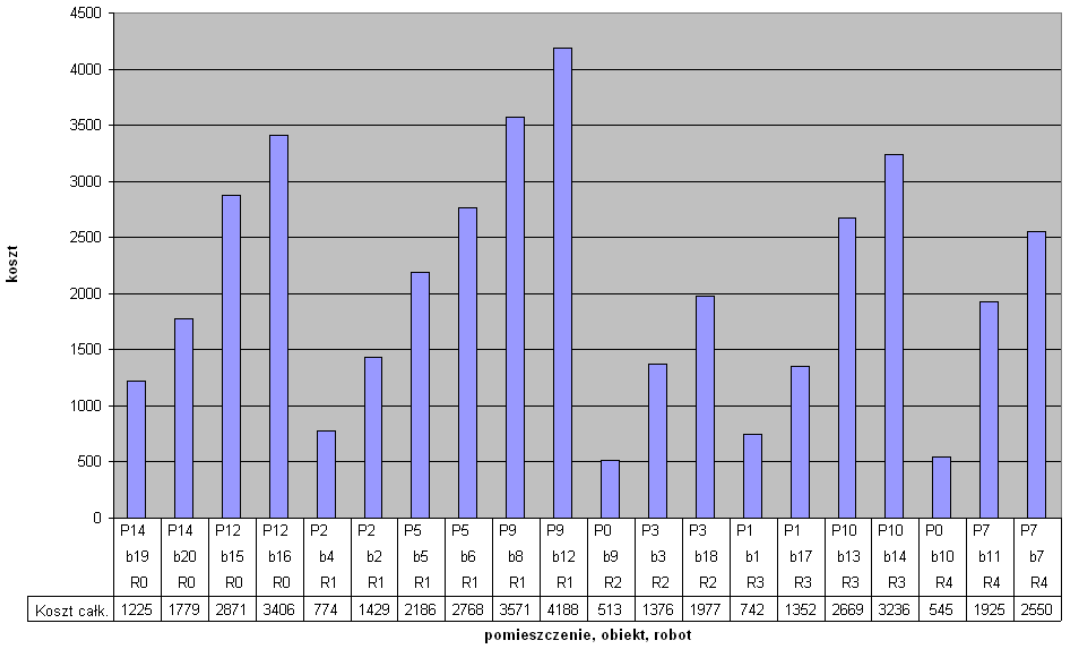
Rysunek 4.22: Mapa typu *Amfilada*, z rozproszonymi poszukiwanymi obiektami charakteryzująca się umieszczeniem pomieszczeń w stylu „jedno za drugim";

pomieszczenie P1. W tym czasie robot R0 sprawdził pomieszczenie P2. Robot R4 po zakończeniu sprawdzania pomieszczenia P1 udał się do pomieszczenia P14 i rozpoczął sprawdanie pomieszczeń począwszy od P14. Zestawienie przeszukanych pomieszczeń i znalezionych obiektów przedstawiono w tabeli 4.3. Widać z niej, że stosunkowo szybko znalezione zostały obiekty znajdujące się na końcu gałęzi, natomiast najdłużej trwało znalezienie obiektów w pomieszczeniu P10 i P9.



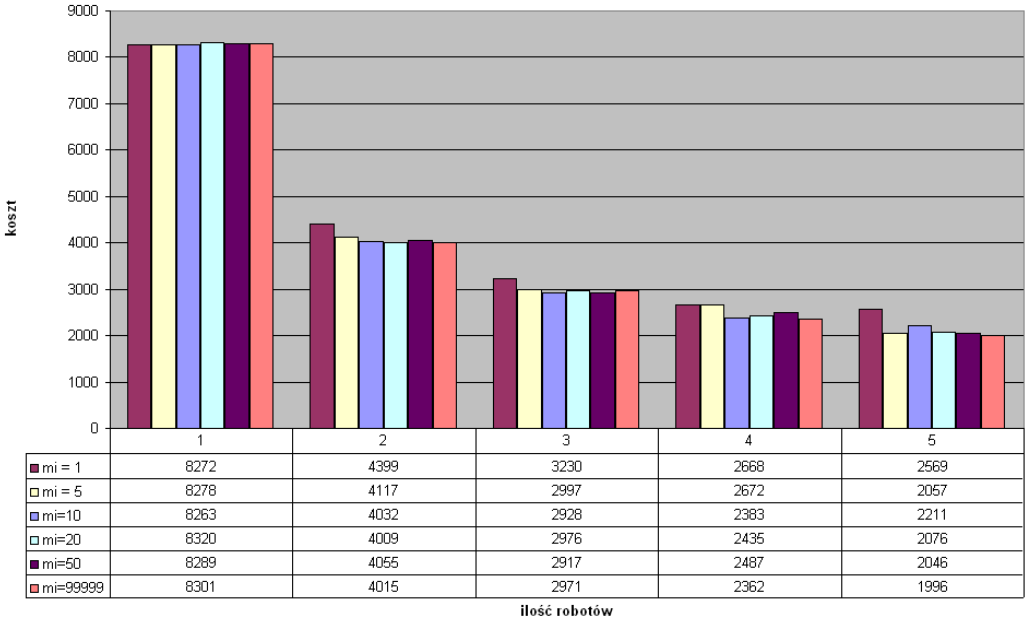
	P2 b4 R0	P2 b2 R0	P7 b7 R0	P7 b11 R0	P9 b8 R0	P9 b12 R0	P12 b16 R0	P12 b15 R0	P0 b10 R1	P3 b3 R1	P5 b6 R1	P5 b5 R1	P10 b13 R1	P0 b9 R2	P1 b1 R3	P3 b18 R3	P1 b17 R4	P14 b19 R4	P14 b20 R4	P10 b14 R4	P10 b13 R4
Koszt całk.	1206	1788	2557	3181	4122	4747	5496	6015	510	1544	3494	4064	4919	510	612	1642	811	2090	2665	3960	4504

Rysunek 4.23: Koszt całkowity znalezienia obiektów; mapa typu *Amfilada*, obiekty rozproszone, μ=1, grupa 5 robotów - jeden z przypadków;



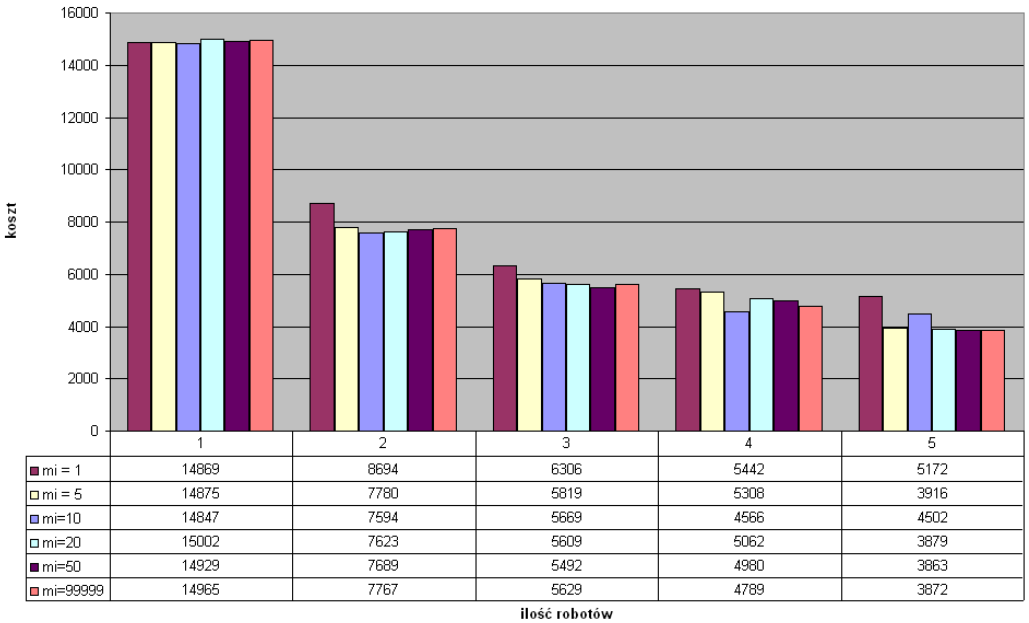
	P14 b19 R0	P14 b20 R0	P12 b15 R0	P12 b16 R0	P2 b4 R1	P2 b2 R1	P5 b5 R1	P5 b6 R1	P9 b8 R1	P9 b12 R1	P0 b9 R2	P3 b3 R2	P3 b18 R2	P1 b1 R3	P1 b17 R3	P10 b13 R3	P10 b14 R3	P0 b10 R4	P7 b11 R4	P7 b7 R4
Koszt całk.	1225	1779	2871	3406	774	1429	2186	2768	3571	4188	513	1376	1977	742	1352	2669	3236	545	1925	2550

Rysunek 4.24: Koszt całkowity znalezienia obiektów; mapa typu *Amfilada*, obiekty rozproszone, $\mu = \infty$, grupa 5 robotów - jeden z przypadków;



	1	2	3	4	5
mi = 1	8272	4399	3230	2668	2569
mi = 5	8278	4117	2997	2672	2057
mi=10	8263	4032	2928	2383	2211
mi=20	8320	4009	2976	2435	2076
mi=50	8289	4055	2917	2487	2046
mi=99999	8301	4015	2971	2362	1996

Rysunek 4.25: Średni koszt dojazdu do poszukiwanego obiektu w mapie typu *Amfilada*, obiekty rozproszone, dla różnej ilości robotów i różnego współczynnika μ



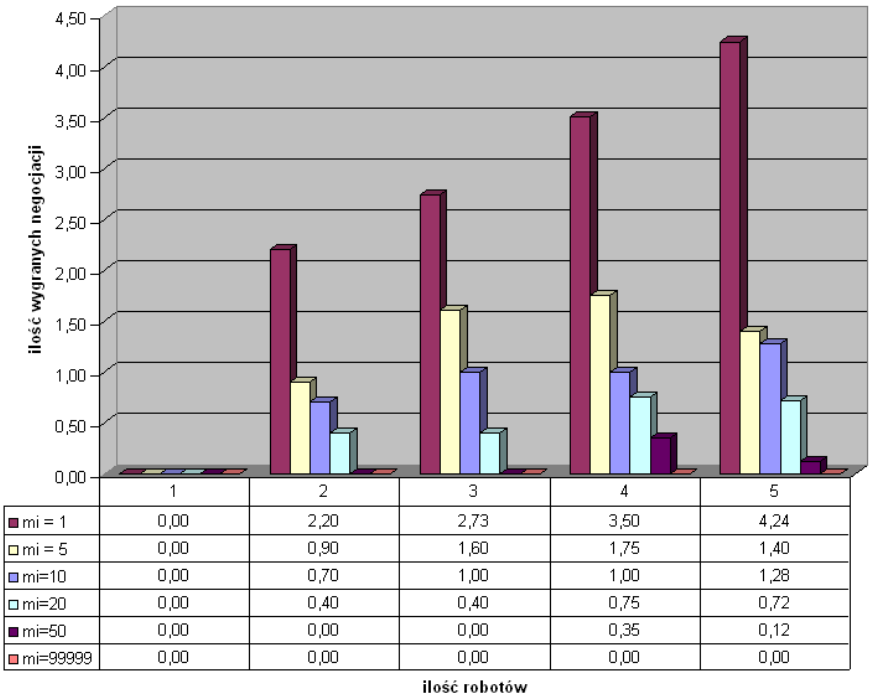
	1	2	3	4	5
mi = 1	14869	8694	6306	5442	5172
mi = 5	14875	7780	5819	5308	3916
mi=10	14847	7594	5669	4566	4502
mi=20	15002	7623	5609	5062	3879
mi=50	14929	7689	5492	4980	3863
mi=99999	14965	7767	5629	4789	3872

Rysunek 4.26: Koszt dojazdu do ostatniego poszukiwanego obiektu w mapie typu *Amfilada*, obiekty rozproszone, dla różnej ilości robotów i różnego współczynnika μ;

L.p.	Robot	Obiekt	Pomieszczenie	Koszt
1	R1	b10	P0	510
2	R2	b9	P0	510
3	R3	b1	P1	612
4	R4	b17	P1	811
5	R0	b4	P2	1206
6	R1	b3	P3	1544
7	R3	b18	P3	1642
8	R0	b2	P2	1788
9	R4	b19	P14	2090
10	R0	b7	P7	2557
11	R4	b20	P14	2665
12	R0	b11	P7	3181
13	R1	b6	P5	3494
14	R4	b14	P10	3960
15	R1	b5	P5	4064
16	R0	b8	P9	4122
17	R4	b13	P10	4504
18	R0	b12	P9	4747
19	R0	b16	P12	5496
20	R0	b15	P12	6015

Tablica 4.3: Kolejność znalezienia obiektów w mapie typu *Amfilada* dla grupy 5 robotów, μ=1 - jeden z przypadków;

Dla $\mu = \infty$, czyli w grupie niewykorzystującej mechanizmu negocjacji, robot R0 jako pierwszy zarezerwował pomieszczenie P14, oraz całą gałąź. Pozostałe roboty z powodu braku niezarezerwowanych pomieszczeń, rozpoczęły sprawdzanie najbliższych wolnych pomieszczeń. Roboty rozpoczęły sprawdzanie następujących pomieszczeń: R4 i R2 - pomieszczenie P0, robot R3 - P1, robot R2 - P3. Po zakończeniu sprawdzania tych pomieszczeń, roboty sprawdziły najbliższe niesprawdzone pomieszczenia. Rysunek 4.24 dokładnie pokazuje, które roboty znalazły które obiekty i jaki był koszt ich znalezienia.



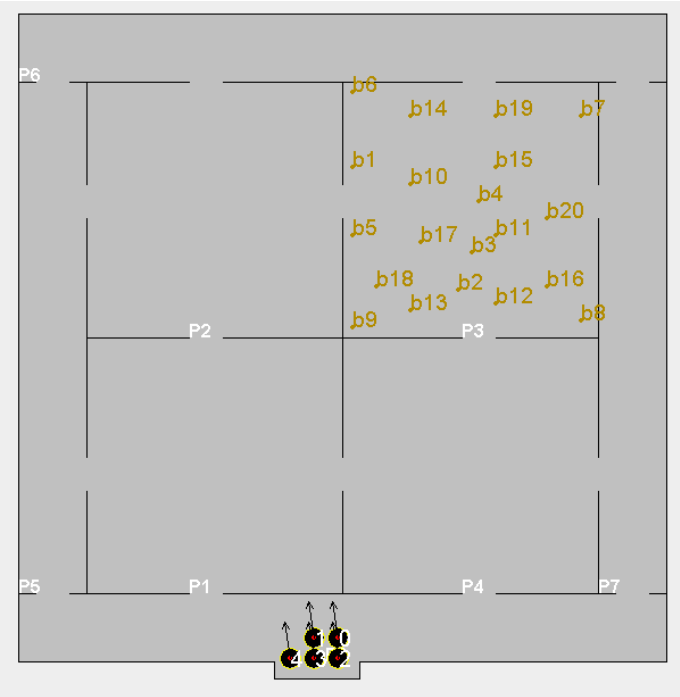
	1	2	3	4	5
mi = 1	0,00	2,20	2,73	3,50	4,24
mi = 5	0,00	0,90	1,60	1,75	1,40
mi=10	0,00	0,70	1,00	1,00	1,28
mi=20	0,00	0,40	0,40	0,75	0,72
mi=50	0,00	0,00	0,00	0,35	0,12
mi=99999	0,00	0,00	0,00	0,00	0,00

Rysunek 4.27: Średnia ilość wygranych negocjacji przypadających na jednego robota w mapie typu *Amfilada*, obiekty rozproszone, dla różnej ilości robotów i różnego współczynnika μ;

4.5 Mapa typu *Każdy-z-każdym* ze skupionymi poszukiwanymi obiektami

Mapa typu *Każdy-z-każdym* charakteryzuje się takim układem pomieszczeń, w którym są one skupione blisko siebie i istnieje możliwość przejazdu do każdego sąsiedniego pomieszczenia (rys.4.28). Cechą charakterystyczną mapy jest stosunkowo skomplikowany graf połączeń pomiędzy pomieszczeniami. Ponieważ roboty eksplorując graf, rezerwują kolejne węzły grafu (czyli pomieszczenia), graf może być przedstawiony w postaci drzewa. Rysunki 4.29 i 4.30 przedstawiają po jednym z przeprowadzonych eksperymentów dla $\mu = 1$ i $\mu = \infty$. Widać z nich, że w przypadku wykorzystującym mechanizm negocjacji w stopniu maksymanlym ($\mu = 1$) koszt znalezienia poszukiwanego obiektu nie przekracza 4000 kroków, podczas gdy mechanizm bez negocjacji ($\mu = \infty$) 8 z 20 obiektów znajduje w czasie dłuższym niż 4000 kroków. Dzięki temu, że graf opisujący ten rodzaj mapy ma wiele krawędzi, obiekty zostały znalezione przez dwa roboty a nie przez jednego jak by to wynikało z opisu systemu (dla $\mu = \infty$). Wynika to z faktu, że wszystkie pomieszczenia są przejściowe i robot będący w trybie dojazdu do pomieszczenia, zmienia swój cel przejeżdzając przez pomieszczenie w którym zostały znalezione obiekty. Z rys.4.30 można wywnioskować, że robot R1 przejeżdzał przez pomieszczenie P3 w czasie

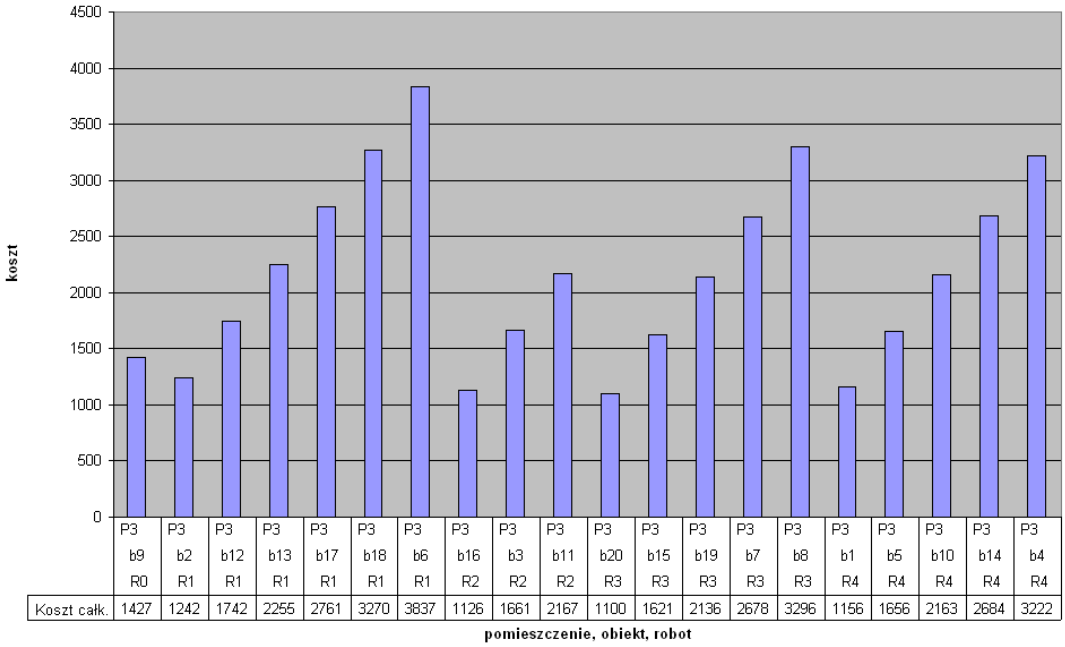
gdy robot R2 znalazł obiekt b20 w tym pomieszczeniu i rozpoczął negocjacje. Koszt dojazdu do pomieszczenia przez które robot przejeżdża w chwili dokonania pomiaru wynosi 0, w związku z czym nierówność 3.2 jest spełniona mimo przyjętego współczynnika $\mu = \infty$ (numerycznie bardzo duża liczba). Takie zdarzenie miało miejsce w grupe 5 robotów i $\mu = \infty$. W przeciwnym przypadku, można oczekiwać, że koszt znalezienia ostatniego obiektu byłby około dwa razy większy.



Rysunek 4.28: Mapa typu *Każdy-z-Każdym*, ze skupionymi poszukiwanymi obiektami;

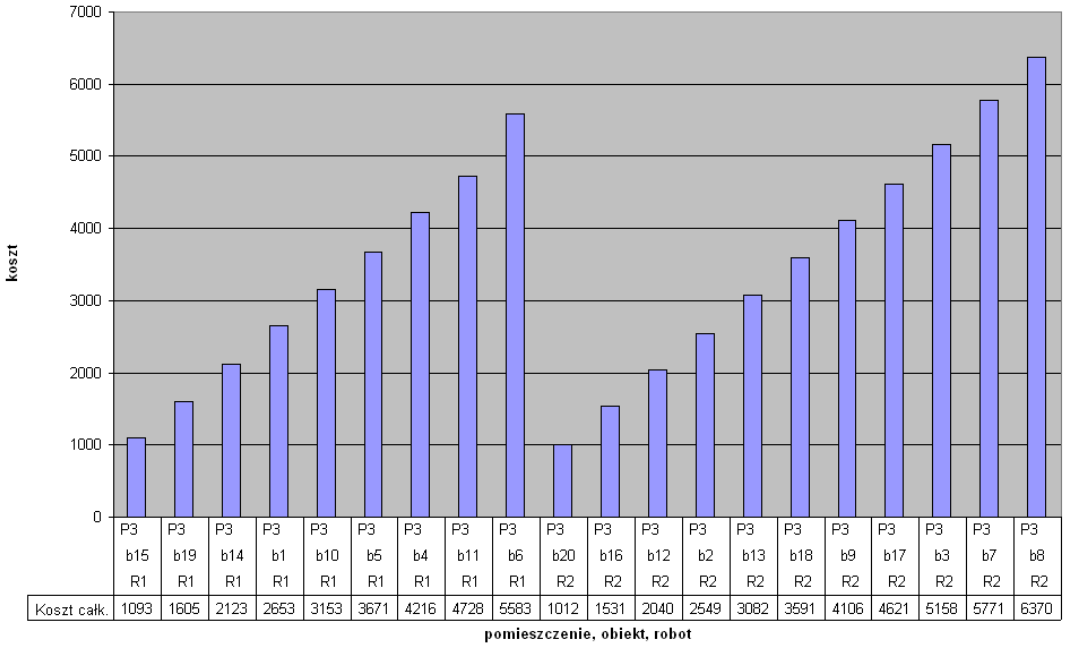
Rysunki 4.31 i 4.32 przedstawiają średni i maksymalny koszt dojazdu do poszukiwanego obiektu. Z rysunków tych wynika, że zastosowanie mechanizmu aukcji zdecydowanie poprawia skuteczność zespołu w każdym przypadku. Dla zespołu 2 robotów, zastosowanie mechanizmu negocjacji z dowolnym z badanych parametrów μ, poprawia skuteczność zespołu o około 35%. Dla grupy składającej się z 3 robotów, najlepsze efekty daje zastosowanie jak najmniejszego parametru μ. Dla μ=1 i μ=5 średni koszt znalezienia obiektu wynosił około 3000 kroków, dla $\mu = 10, 20, 50$ - około 10% więcej, natomiast grupa niewykorzystująca mechanizmu negocjacji w ogóle - około 4600 kroków. W grupie składającej się z 4 robotów widać różnicę w skuteczności zespołu dla różnych wartości współczynnika μ. Dla μ=50, czyli w przypadku bardzo słabo wykorzystywanego mechanizmu negocjacji, wynik jest porównywalny z grupą niewykorzystującą negocjacji w ogóle, natomiast dla wartości $\mu \leq 20$, wyniki są o około 25% lepsze. Podobna sytuacja jest w grupie 5 robotów. Dla μ=50 średni koszt znalezienia obiektu jest o około 23% mniejszy niż dla grupy bez negocjacji, natomiast dla jeszcze mniejszych wartości μ, średni koszt

znalezienia obiektu jest o około 34% mniejszy niż dla $\mu = \infty$.



Rysunek 4.29: Koszt całkowity znalezienia obiektów; mapa typu *Kazdy-z-kazdym*, obiekty skupione, μ=1, grupa 5 robotów - jeden z przypadków;

Z rysunku 4.33 wynika, że w grupach, w których skutecznie wykorzystano zaawansowany mechanizm negocjacji, średnia ilość wygranych aukcji przypadających na jednego robota, waha się w przedziale 0,5 - 0,9. W zespołach, w których wartość ta jest niższa, koszt znalezienia robota jest wyższy. W porównaniu z innymi prezentowanymi mapami stosunkowo niska jest najwyższa średnia ilość wygranych aucji (0,9, wobec około 1,7 dla innych map ze skupionymi obiektami). Wynika to z faktu, że w innych prezentowanych mapach wszystkie skupione obiekty były rozmieszczone w dwóch pomieszczeniach, natomiast w niniejszej mapie, wszystkie obiekty są skupione w jednym pomieszczeniu.
W grupie składającej się z dwóch robotów wszystkie wyniki dla $\mu \leq 50$ są identyczne. Wynika to z faktu, że jeden robot znalazł poszukiwane obiety, drugi robot wygrał aukcję i roboty wspólnie przeszukiwały pomieszczenie z obiektami. To tłumaczy, dlaczego na rys.4.33 średnia ilość wygranych negocjacji wynosi 0,5 dla wszystkich wartości $\mu \leq 50$. Dla μ=50, ze wzrostem liczby robotów w grupie średnia ilość wygranich aukcji przypadających na jednego robota zaczęła spadać, ponieważ liczba robotów w zespole rośnie a tylko 1 robot wygrał 1 aukcję. Dla innych wartości μ więcej robotów wygrało aukcję i wzrost liczby robotów w zespole spowodował wzrost średniej liczby wygranych aukcji.

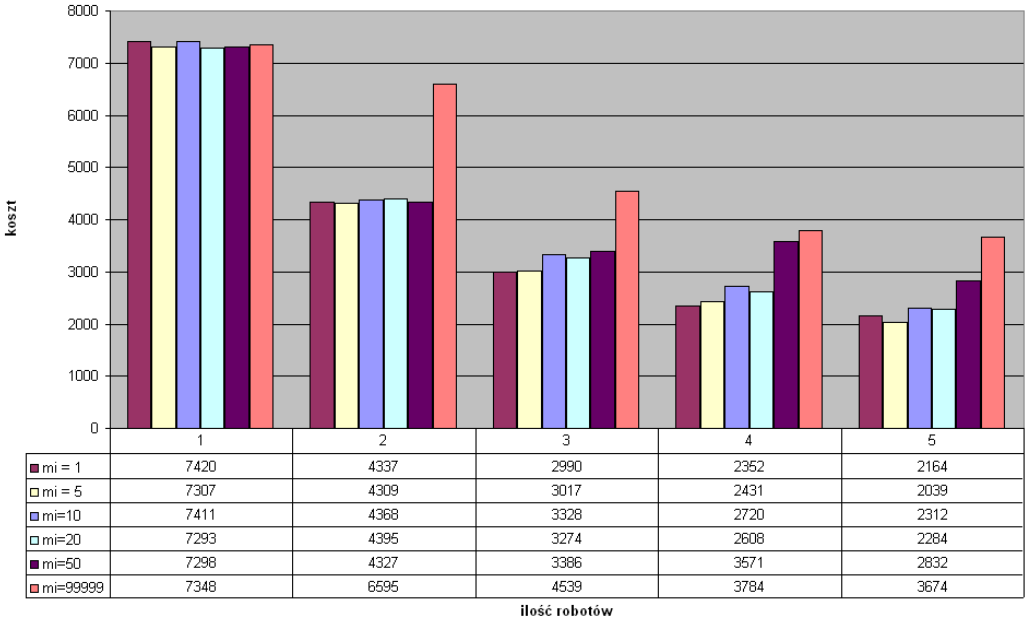


Rysunek 4.30: Koszt całkowity znalezienia obiektów; mapa typu *Każdy-z-Każdym*, obiekty skupione, $\mu = \infty$, grupa 5 robotów - jeden z przypadków;

4.6 Mapa typu *Każdy-z-każdym* z rozproszonymi poszukiwanymi obiektami

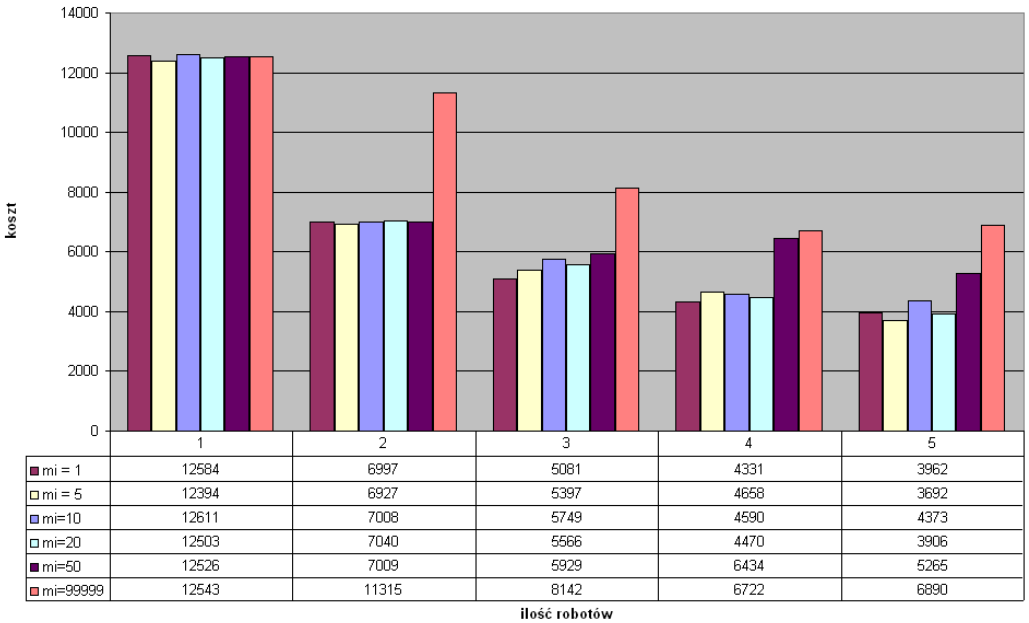
Z rysunku 4.34 widać, że w rozpatrywanym przypadku obiekty są rozrzucone w miarę równomiernie we wszystkich pomieszczeniach. Można się spodziewać, że najbardziej efektywny byłby taki rozdział pracy, w którym każdy robot niezależnie eksploruje inne pomieszczenie. Rysunki 4.35 i 4.36 przedstawiają po jednym eksperymentcie przeprowadzonym dla skrajnych wartości współczynnika chęci współpracy μ. Z rysunków tych wynika, że dla μ=1 (czyli pełna współpraca robotów) maksymalny koszt znalezienia ostatniego obiektu wynosi 5200 kroków, podczas gdy dla $\mu = \infty$ - 3800 kroków, czyli zastosowanie mechanizmu negocjacji robotów w tym wypadku pogarsza skuteczność całego zespołu o około 36%. Rozkład ilości robotów w zależności od współczynnika μ przedstawia tabela 4.4. Z tabeli tej wynika, że dla $\mu = \infty$, rozkład ilości znalezionych robotów jest prawie idealny (średnia znalezionych obiektów, przypadających na jednego robota wynosi 4), a dla μ=1, rozkład jest nieco gorszy. Uwzględniając ten fakt, oraz koszt wydany na zmianę celu w związku z wygraną aukcją, widać, dlaczego maksymalny koszt znalezienia obiektu jest wyższy w przypadku μ=1, niż $\mu = \infty$.

Rysunki 4.37 i 4.38 pokazują średni i maksymalny koszt znalezienia poszukiwanych obiektów. Z rysunków tych widać, że dla grupy 2 i 3 robotów i $\mu \geq$10, skuteczność grupy nie zmienia się (średni koszt wynosi około 4000 kroków dla grupy 2 robotów i 3000 dla



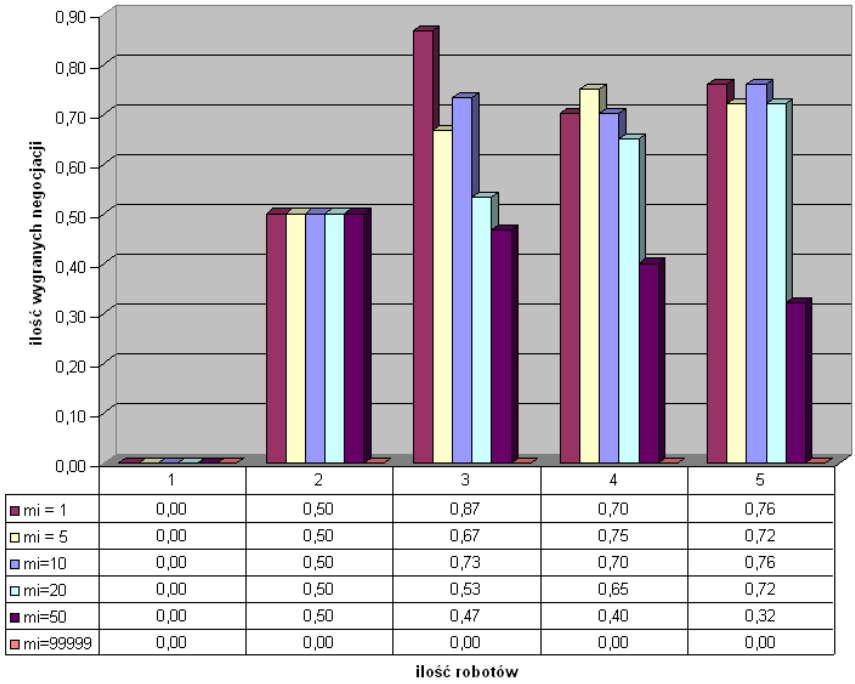
	1	2	3	4	5
mi = 1	7420	4337	2990	2352	2164
mi = 5	7307	4309	3017	2431	2039
mi=10	7411	4368	3328	2720	2312
mi=20	7293	4395	3274	2608	2284
mi=50	7298	4327	3386	3571	2832
mi=99999	7348	6595	4539	3784	3674

Rysunek 4.31: Średni koszt dojazdu do poszukiwanego obiektu w mapie typu *Każdy-z-Każdym*, obiekty skupione, dla różnej ilości robotów i różnego współczynnika μ



	1	2	3	4	5
mi = 1	12584	6997	5081	4331	3962
mi = 5	12394	6927	5397	4658	3692
mi=10	12611	7008	5749	4590	4373
mi=20	12503	7040	5566	4470	3906
mi=50	12526	7009	5929	6434	5265
mi=99999	12543	11315	8142	6722	6890

Rysunek 4.32: Koszt dojazdu do ostatniego poszukiwanego obiektu w mapie typu *Każdy-z-Każdym*, obiekty skupione, dla różnej ilości robotów i różnego współczynnika μ;



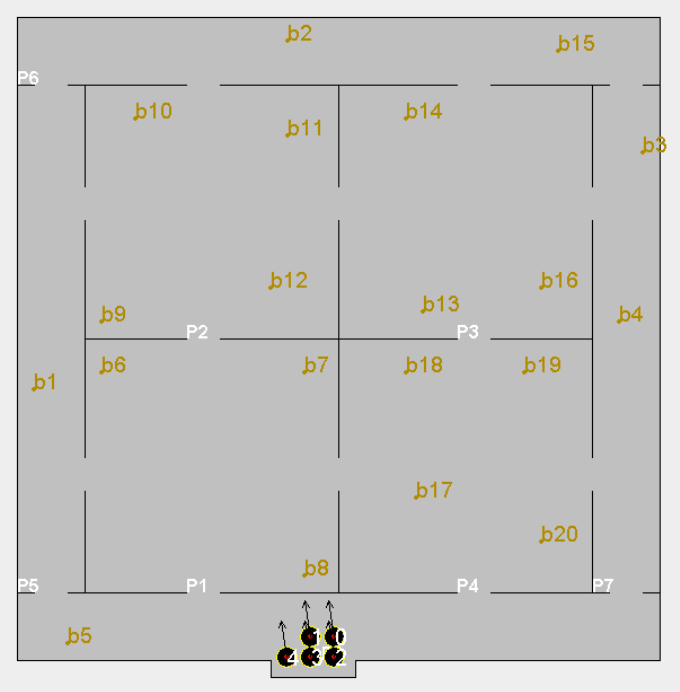
	1	2	3	4	5
mi = 1	0,00	0,50	0,87	0,70	0,76
mi = 5	0,00	0,50	0,67	0,75	0,72
mi=10	0,00	0,50	0,73	0,70	0,76
mi=20	0,00	0,50	0,53	0,65	0,72
mi=50	0,00	0,50	0,47	0,40	0,32
mi=99999	0,00	0,00	0,00	0,00	0,00

Rysunek 4.33: Średnia ilość wygranych negocjacji przypadających na jednego robota w mapie typu *Każdy-z-Każdym*, obiekty skupione, dla różnej ilości robotów i różnego współczynnika μ;

Robot	μ=1	$\mu = \infty$
R0	6	5
R1	2	4
R2	6	4
R3	3	4
R4	3	3

Tablica 4.4: Rozkład znalezionych obiektów, dla mapy typu *Każdy-z-każdym*, obiekty rozporszone, dla skrajnych wartości współczynnika chęci współpracy μ - porównanie po jednym z eksperymentów;

grupy 3 robotów), natomiast dla mniejszych wartości μ - nieznacznie się pogarsza (dla μ=1, średni koszt - około 5000 kroków dla grupy 2 robotów i 3500 dla 3 robotów). W zespołach 4 i 5 robotów zmniejszanie współczynnika μ powoduje stopniowy wzrost kosztu (19% dla 4 i 29% dla 5 robotów).
Z rysunku 4.39, widać wpływ zmiany współczynnika μ na ilość wygranych aukcji a tym samym na zachowanie się robota. Dla μ=1 średnia ilość wygranych aukcji waha się w przedziale 2,4 - 3,1, podczas gdy dla μ=5 ten przedział wynosi 0,9 - 1,4. Dla wyższych

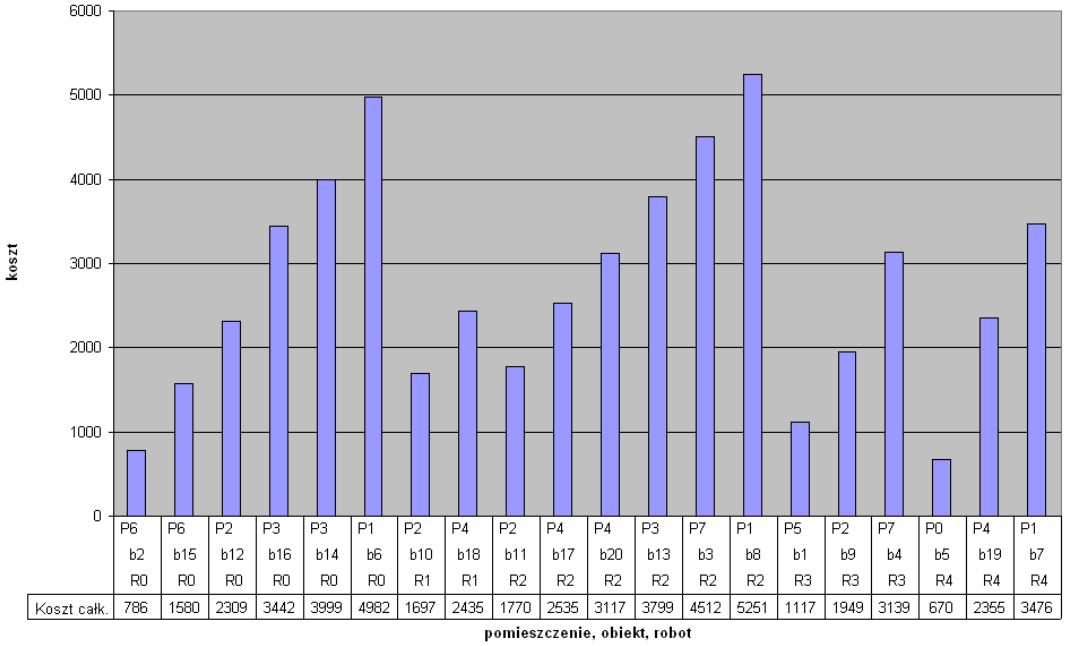


Rysunek 4.34: Mapa typu *Każdy-z-Każdym*, z rozproszonymi poszukiwanymi obiektami;

wartości μ średni koszt wygranych aukcji nie przekracza 1,0.
W rozpatrywanym przypadku, w każdym pomieszczeniu są poszukiwane obiekty. W związku z tym, tylko te roboty uczestniczą w aukcji, które nie znalazły jeszcze obiektu w swoim pomieszczeniu, gdyż w przeciwnym razie robot nie byłby zainteresowany aukcją. Część obiektów została znaleziona szybciej w wyniku zastosowania mechanizmu aukcji, kosztem pozostałych obiektów (na przykład obiekty b9 i b10 zostały znalezione szybciej w przypadku μ=1, podczas gdy zdecydowana więszkość obiektów została znaleziona szybciej w przypadku $\mu = \infty$ - rysunki 4.35 i 4.36).
Porównując średnie ilości wygranych negocjacji dla każdej z map dla obiektów skupionych (rysunki 4.9, 4.21, 4.33) widać, że w mapie typu „Każdy-z-każdym" średnia ilość wygranych negocjacji jest mniejsza niż pozostałych przypadkach, oraz dla małych i średnich wartości μ w grupie 4 i 5 robotów, ilość wygranych negocjacji nie zmienia się, podczas gdy dla innych map ilości wygranych negocjacji są większe a zwiększanie współczynnika μ powoduje spadek ilości wygranych aukcji. Są dwa powody takiej sytuacji:

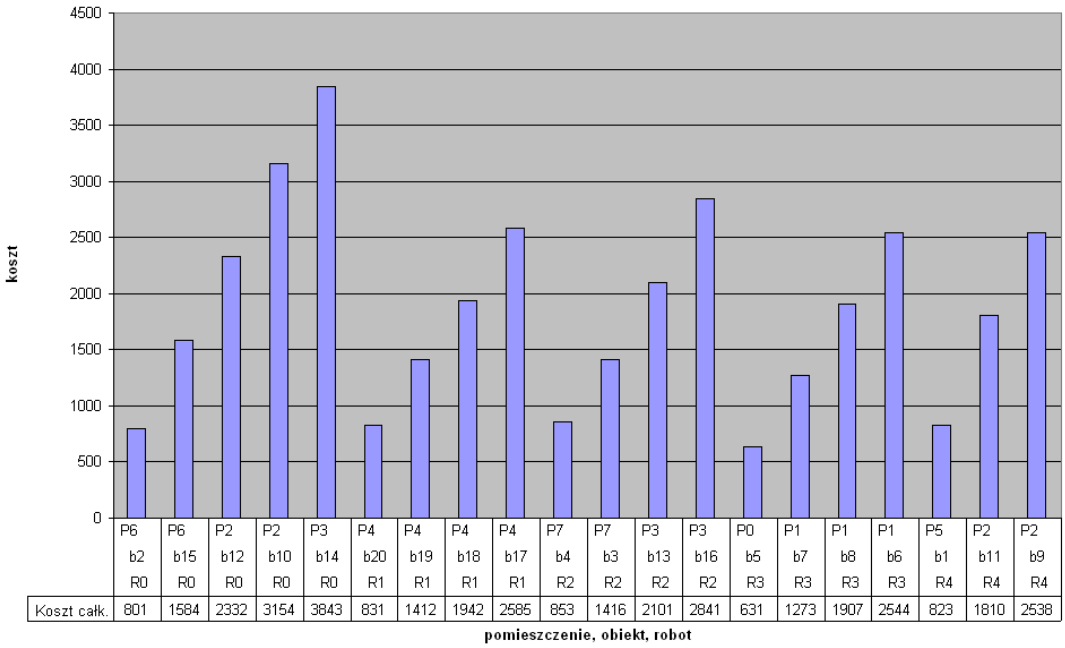
- W mapie typu "każdy-z-każdym" obiekty są skupione w jednym pomieszczeniu a nie w dwóch jak to ma miejsce w pozostałych mapach. W związku z tym każdy z robotów może wygrać tylko jedną aukcję (poza robotem, który pierwszy znalazł obiekt). W pozostałych mapach roboty mogą wygrać aukcję dwa razy;

- odległości między pomieszczeniami są mniejsze niż w innych mapach. W związku z tym koszt dojazdu do pomieszczenia, w którym zainicjowano aukcję, jest stosunkowo



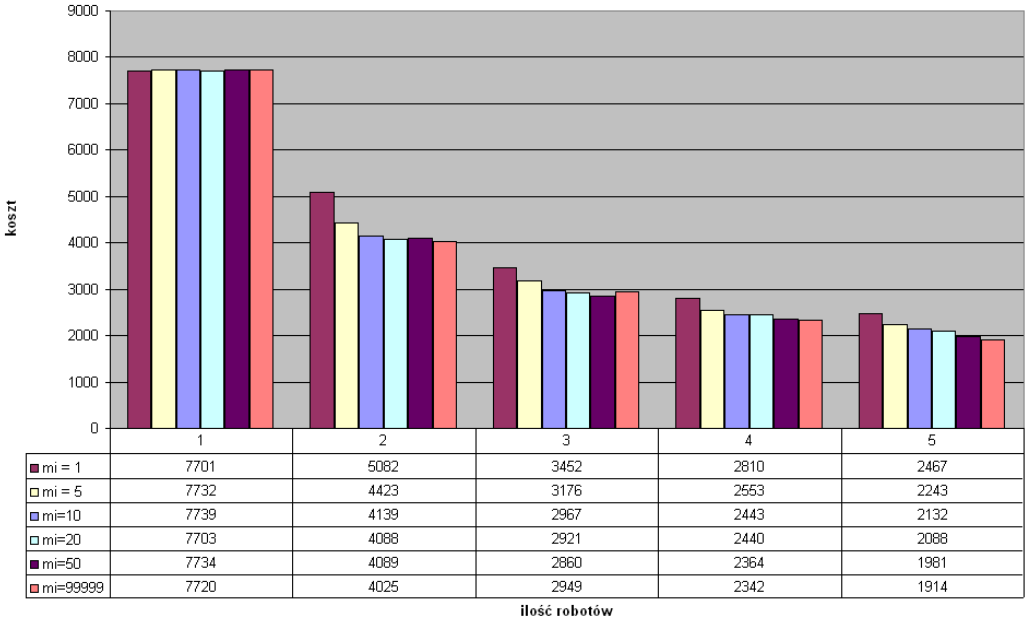
	P6 b2 R0	P6 b15 R0	P2 b12 R0	P3 b16 R0	P3 b14 R0	P1 b6 R0	P2 b10 R1	P4 b18 R1	P2 b11 R2	P4 b17 R2	P4 b20 R2	P3 b13 R2	P7 b3 R2	P1 b8 R2	P5 b1 R3	P2 b9 R3	P7 b4 R3	P0 b5 R4	P4 b19 R4	P1 b7 R4
Koszt całk.	786	1580	2309	3442	3999	4982	1697	2435	1770	2535	3117	3799	4512	5251	1117	1949	3139	670	2355	3476

Rysunek 4.35: Koszt całkowity znalezienia obiektów; mapa typu *Kazdy-z-kazdym*, obiekty rozproszone, μ=1, grupa 5 robotów - jeden z przypadków;



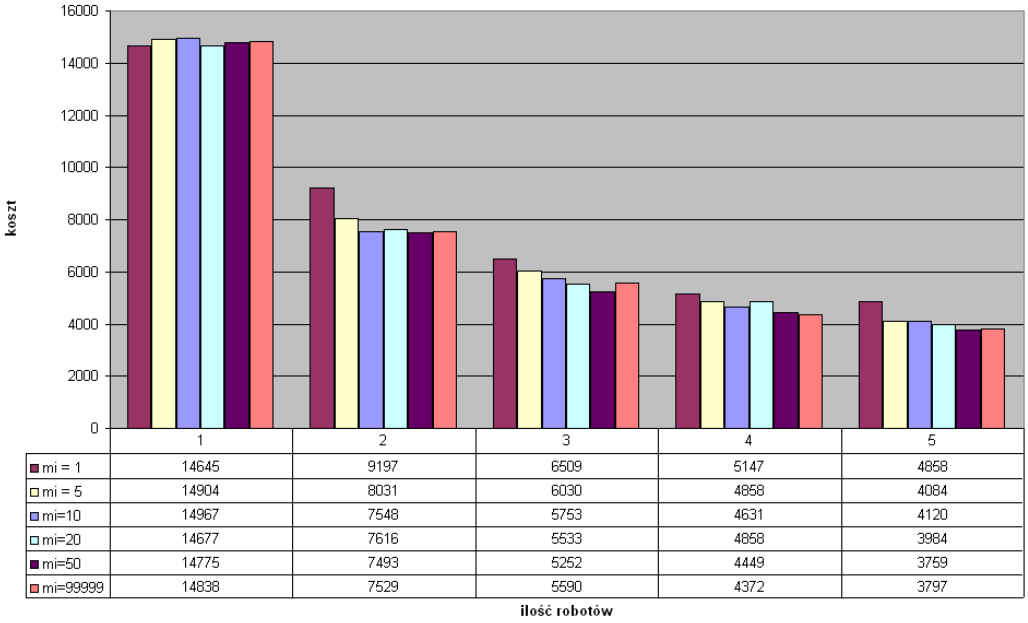
	P6 b2 R0	P6 b15 R0	P2 b12 R0	P2 b10 R0	P3 b14 R0	P4 b20 R1	P4 b19 R1	P4 b18 R1	P4 b17 R1	P7 b4 R2	P7 b3 R2	P3 b13 R2	P3 b16 R2	P0 b5 R3	P1 b7 R3	P1 b8 R3	P1 b6 R3	P5 b1 R4	P2 b11 R4	P2 b9 R4
Koszt całk.	801	1584	2332	3154	3843	831	1412	1942	2585	853	1416	2101	2841	631	1273	1907	2544	823	1810	2538

Rysunek 4.36: Koszt całkowity znalezienia obiektów; mapa typu *Każdy-z-Każdym*, obiekty rozproszone, $\mu = \infty$, grupa 5 robotów - jeden z przypadków;



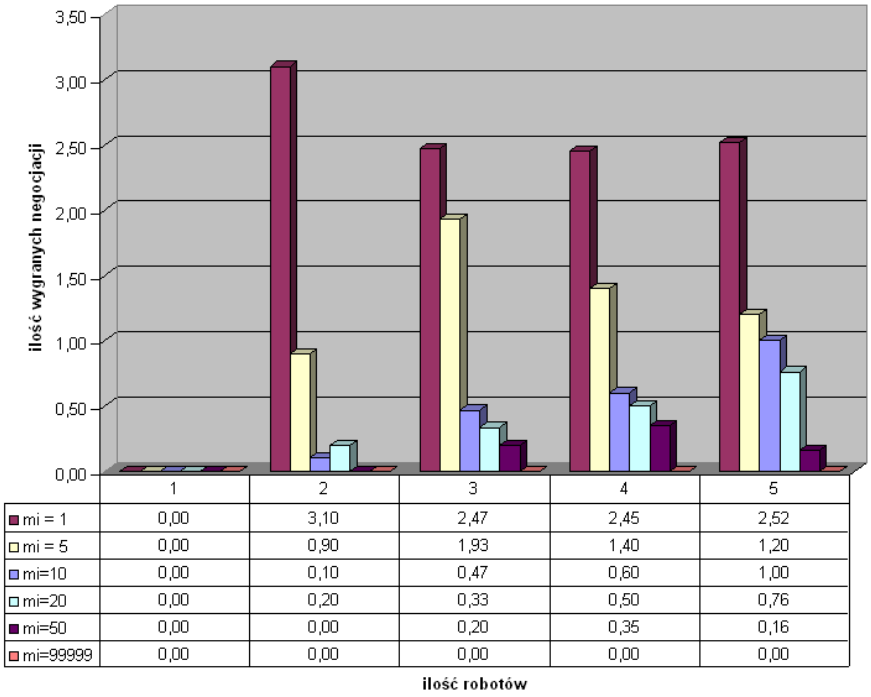
	1	2	3	4	5
mi = 1	7701	5082	3452	2810	2467
mi = 5	7732	4423	3176	2553	2243
mi=10	7739	4139	2967	2443	2132
mi=20	7703	4088	2921	2440	2088
mi=50	7734	4089	2860	2364	1981
mi=99999	7720	4025	2949	2342	1914

Rysunek 4.37: Średni koszt dojazdu do poszukiwanego obiektu w mapie typu *Każdy-z-Każdym*, obiekty rozproszone, dla różnej ilości robotów i różnego współczynnika μ



	1	2	3	4	5
mi = 1	14645	9197	6509	5147	4858
mi = 5	14904	8031	6030	4858	4084
mi=10	14967	7548	5753	4631	4120
mi=20	14677	7616	5533	4858	3984
mi=50	14775	7493	5252	4449	3759
mi=99999	14838	7529	5590	4372	3797

Rysunek 4.38: Koszt dojazdu do ostatniego poszukiwanego obiektu w mapie typu *Każdy-z-Każdym*, obiekty rozproszone, dla różnej ilości robotów i różnego współczynnika μ;



	1	2	3	4	5
mi = 1	0,00	3,10	2,47	2,45	2,52
mi = 5	0,00	0,90	1,93	1,40	1,20
mi=10	0,00	0,10	0,47	0,60	1,00
mi=20	0,00	0,20	0,33	0,50	0,76
mi=50	0,00	0,00	0,20	0,35	0,16
mi=99999	0,00	0,00	0,00	0,00	0,00

Rysunek 4.39: Średnia ilość wygranych negocjacji przypadających na jednego robota w mapie typu *Każdy-z-Każdym*, obiekty rozproszone, dla różnej ilości robotów i różnego współczynnika μ;

mały i nawet dla średnich wartości μ nierówność 3.2 jest spełniona.

Rozdział 5

Opis symulatora współpracy robotów

5.1 Cel

Opisywany symulator może być wykorzystany w badaniu zachowań pojedynczych robotów, oraz zespołów wielorobotowych. Badany robot może być wyposażony w dowolną ilość sensorów, czujników dotykowych, może rozpoznawać inne roboty i znaczniki. Logowanie wymiany komunikatów pomiędzy robotami a serwerem umożliwia późniejszą analizę offline.

5.2 Opis symulatora

Opisywany serwer został utworzony w języku programowania „Java". Idea serwera została oparta o istniejący symulator gry piłkę nożną - Soccerserver. (www.robocup.org). Umożliwia on rozgrywanie wirtualnych meczy pomiędzy drużynami. Każda drużyna składa się z 11. graczy - agentów. Symulator składa się z serwera, symulującego środowisko zewnętrzne, oraz klientów symulujących pracę robotów mobilnych. Do serwera jest wczytywana mapa zawierająca ściany, przeszkody i znaczniki. Komunikacja pomiędzy serwerem a klientami jest realizowana za pomocą połączeń TCP/IP. Uruchomienie symulatora polega na uruchomieniu serwera środowiska (serwera TCP), wybraniu mapy otoczenia, oraz uruchomienia robota lub robotów (klientów TCP). Co zadany czas (domyślnie 100ms) serwer wysyła do klientów informację na temat otoczenia robota: stany czujników, odebrane komunikaty od innych robotów, stany czujników dotykowych (zderzaków), informacja o widzianych znacznikach (opcjonalnie), informacja o widzianych robotach (opcjonalnie). Na żądanie robot może wysyłać do serwera informację o aktywowaniu swoich efektorów. Dostępne komendy to: „Krok naprzód" (o zadaną długości), „Krok w tył" (o zadaną długość), obrót (o zadany kąt), krzyknij (dowolny tekst). Jeżeli jeden klient wyśle do serwera komendę, w następnym kroku wszystkie roboty będące w stanie zobaczyć ten ruch lub usłyszeć wysyłaną komendę dostaną od serwera odpowiednią informację. Wszelkie parametry robota takie jak: rozmiary robota, ilość czujników dotykowych, zbliże-

niowych, zasięg czujników, zasięg rozpoznawania innego robota, zasięg rozpoznawania znaczników mogą być ustawiane oddzielnie dla każdego robota. Ponadto roboty posiadają błąd odometrii, którego prawdopodobieństwo pojawienia się jest ustawiane oddzielnie dla każdego robota. Taka elastyczność robota pozwala na analizę dużej ilości przypadków, która wiernie odwzorowuje rzeczywistość.

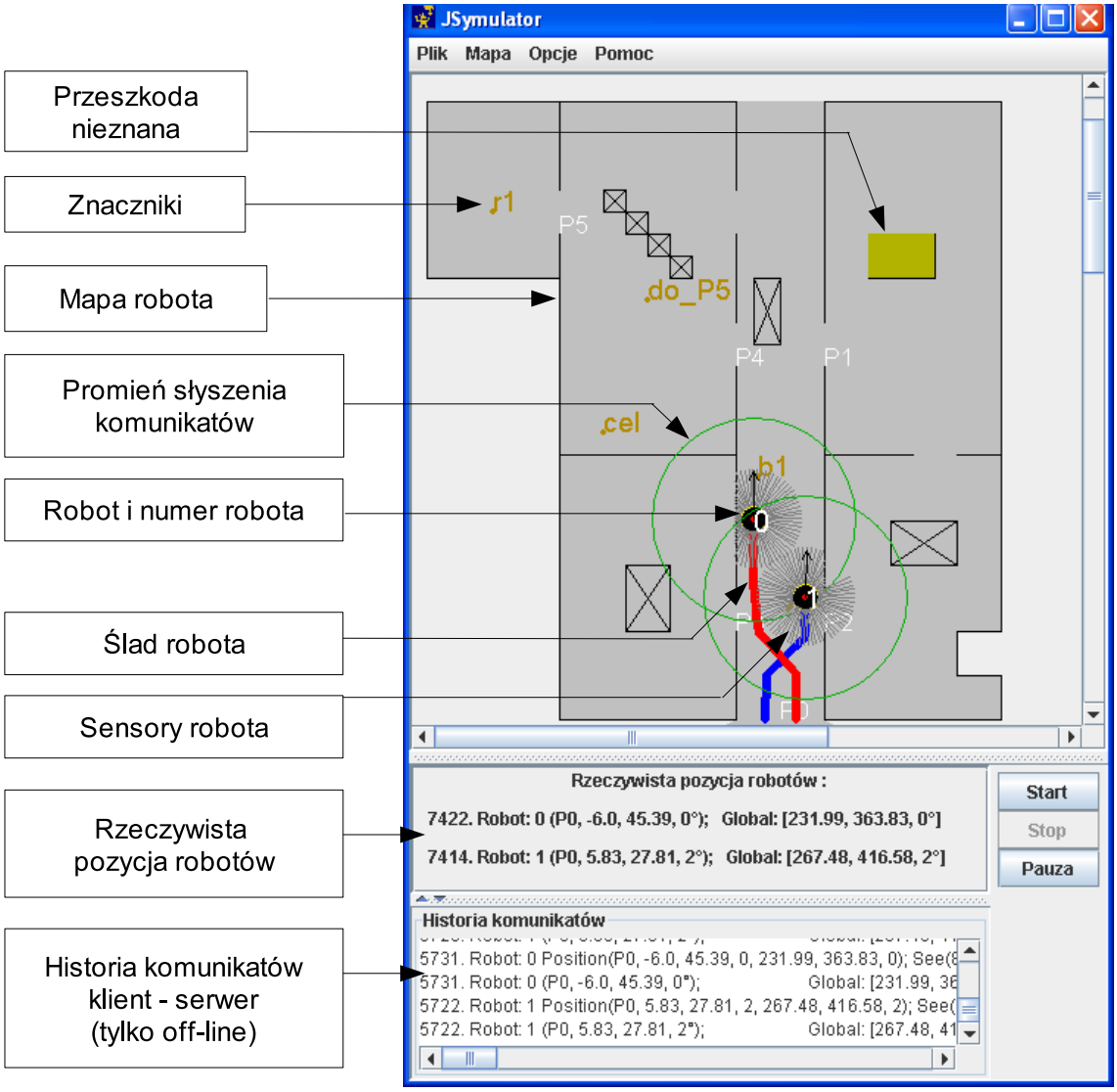
5.3 Serwer środowiska

Serwer został przedstawiony na rys. 5.1. Część górna serwera przedstawia obszar pracy robotów. Widoczna jest mapa otoczenia (wgrana z oddzielnego pliku), wraz za znacznikami i przeszkodami nie należącymi do mapy, roboty (wyświetlony jest robot w skali, numer robota, mogą być wyświetlone linie sonarów robota, oraz promień słyszenia komunikatów). Środkowa część wyświetla rzeczywistą pozycję robota, wraz z numerem wysłanego komunikatu. Trzecia, najniższa część serwera umożliwia wyświetlanie historii komunikatów z robotami i służy do analizy off-line.

5.4 Mapa

Mapa składa się z pomieszczeń i drzwi i jest zapisana w pliku tekstowym. Konfiguracja mapy wymaga podania wszystkich ścian w lokalnych współrzędnych dla danego pomieszczenia. Drzwi określają początek lokalnych współrzędnych (dla pomieszczenia). Przykładowo dla pomieszczenia „P0" zdefiniowano drzwi: (10,85,P1) - co oznacza że w punkcie (10,85) wg współrzędnych z pomieszczenia „P0" są drzwi które prowadzą do pomieszczenia „P1". W pomieszczeniu „P1" zdefiniowano drzwi: d1:(0,5,P0), co oznacza że we współrzędnych (0,5) wg pomieszczenia „P1" są drzwi prowadzące do pomieszczenia „P0". Taki system pozwala jednoznacznie określić zależności pomiędzy lokalnymi układami współrzędnych pomiędzy poszczególnymi pomieszczeniami.
Wszystkie słowa zaczynające się od znaku _ (dolne podkreślenie) są słowami kluczowymi. Najpierw trzeba zdefiniować początek lokalnego układu współrzędnych (tutaj przyjęto punkt (250, 500); można podać współczynnik skalowania mapy (tutaj: 3); można w pliku zapisać rozmiary robota (chociaż nie jest to zalecane), oraz opisać wszystkie pomieszczenia i drzwi.
Mapa przedstawiona w rys. 5.1 jest zapisana następująco:

```
_Poczatek: 250, 500; // początek układu w globalnym ukł. wsp. dla pom.P0.
_Wspolczynnik: 3; // współczynnik przez który jest skalowana mapa
_Promien_robota: 3; // promień robota
_Promien_widzenia: 8; // promień „widzenia" robota
_Promien_slyszenia: 20;// promień „słyszenia" komunikatów
_Promien_widz_rob: 8; // promien widzenia innych robotów
_Promien_widz_zna: 9; // promien widzenia znaczników
```



Rysunek 5.1: Serwer z zaznaczonymi dwoma robotami nie widzącymi się, ale słyszącymi swoje komunikaty;

P0:{(-10,0,-10,20), (-10,30,-10,80), (-10,90,-10,110), (-10,120,-10,140), (10,140,10,90), (10,80,10,30), (10,20,10,0), (20,-5,0,-15), (0,-15,-20,-5)} //ściany w pom. P0
d0:{(-10,25,P3), (-10,85,P4), (-10,115,P4), (10,85,P1), (10,25,P2)} //drzwi w pom. P0
_Przeszkoda_znana:{(P0, -6, 100, 6, 15)}
_Znaczniki: {(b1, P0, -5, 55, 500)}
_Znaczniki: {(r2, P0, 2, 25, 500)}

P1:{(0,0,0,-20), (0,-20,20,-20), (30,-20,40,-20), (40,-20,40,60), (40,60,0,60), (0,60,0,10)}

D1:{(0,5,P0), (25,-20,P2)}
_Przeszkoda_nieznana:{(P1, 10, 30, 15, 10)}

P2:{(0,0,0,-20), (0,-20,40,-20), (40,-20,40,-10), (40,-10,30,-10), (30,-10,30,0), (30,0,40,0), (40,0,40,40), (40,40,30,40), (20,40,0,40), (0,40,0,10)}
_Przeszkoda_znana:{(P2, 15, 25, 15, 10)} //lewy górny róg, szer, wys.
D2:{(0,5,P0), (25,40,P1)}

P3:{(0,0,0,-20), (0,-20,-40,-20), (-40,-20,-40,40), (-40,40,0,40), (0,40,0,10)}
D3:{(0,5,P0)}
_Przeszkoda_znana:{(P3, -25, 15, 10, 15)}

P4:{(0,0,0,-20), (0,-20,-40,-20), (-40,-20,-40,30), (-40,40,-40,60), (-40,60,0,60), (0,60,0,40), (0,30,0,10)}
D4:{(0,5,P0), (0,35,P0), (-40,35,P5)}
_Przeszkoda_znana:{(P4, -30, 40, 5, 5), (P4, -25, 35, 5, 5), (P4, -20, 30, 5, 5), (P4, -15, 25, 5, 5)}
_Znaczniki: {(cel, P4, -30, -15, 500)} //cel
_Znaczniki: {(do_P5, P4, -20, 15, 500)}

P5:{(0,0,0,-10), (0,-10,-30,-10), (-30,-10,-30,30), (-30,30,0,30), (0,30,0,10)}
D5:{(0,5,P4)}
_Znaczniki: {(r1, P5, -15, 5, 500)}

5.5 Klienci TCP/IP

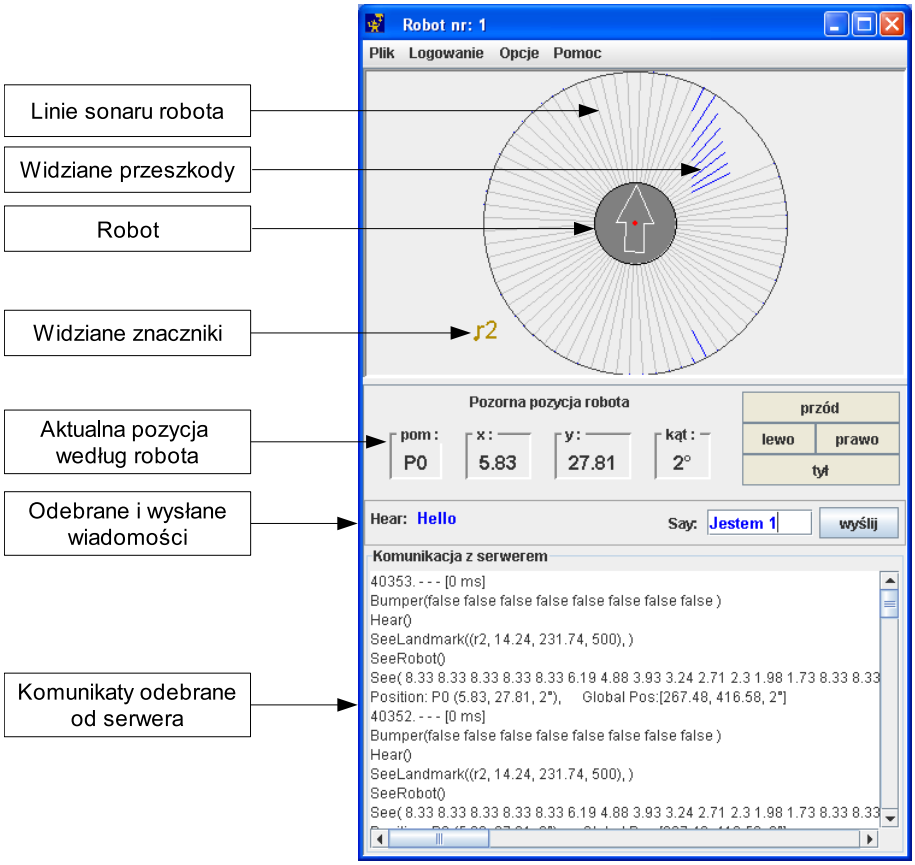
Klient TCP wymienia komunikaty z serwerem. Okno klienta (robota) przedstawione jest na rysunku 5.2. Górna część okna przedstawia graficznie stan sonarów robota. Na rysunku widać, że z prawej strony robot widzi ścianę z drzwiami. Z lewej strony robot widzi znacznik: „r1". Środkowa część okna przedstawia pozorną pozycję robota, oraz ostatnio usłyszany komunikat (tutaj: „Hello"), oraz umożliwia „ręczne" sterowanie robotem (jazda w dowolnym kierunku, oraz wysyłanie wiadomości (tutaj: „Jestem 1"). Pozorna pozycja robota jest to pozycja węglug robota - nieuwzględniająca błędów odometrii. W szczególnym przypadku może być identyczna z rzeczywistą pozycją robota (gdy robot nie ma błędów wynikających z odometrii). Dolna część okna przedstawia komunikaty odbierane z serwera. Można wyświetlać wszystkie, lub tylko wybrane komunikaty (opcja ustawiana w menu). Niżej przedstawiono przykładowe wiadomości robota 0 z rrysunku 5.2.

40353. - - - [0 ms]
Bumper(false false false false false false false false)

Hear()
SeeLandmark((r2, 14.24, 231.74, 500),)
SeeRobot()
See(8.33 8.33 8.33 8.33 8.33 6.19 4.88 3.93 3.24 2.71 2.3 1.98 1.73 8.33 8.33 8.33 8.33
8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 5.89 7.68 8.33 8.33 8.33
8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33
8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33
8.33)
Position: P0 (5.83, 27.81, 2°), Global Pos:[267.48, 416.58, 2°]
40352. - - - [0 ms]
Bumper(false false false false false false false false)
Hear()
SeeLandmark((r2, 14.24, 231.74, 500),)
SeeRobot()
See(8.33 8.33 8.33 8.33 8.33 6.19 4.88 3.93 3.24 2.71 2.3 1.98 1.73 8.33 8.33 8.33 8.33
8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 5.89 7.68 8.33 8.33 8.33
8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33
8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33 8.33
8.33)
Position: P0 (5.83, 27.81, 2°), Global Pos:[267.48, 416.58, 2°]
. . .

Ważniejsze komendy wykorzystane w komunikatach przesyłanych od serwera do klienta są następujące:

- *See* - informacja o przeszkodach widzianych przez czujnik, jeżeli wartość jest mniejsza od maksymalnej. Tutaj maksymalna odległość działania czujnika wynosi 8 [kroków]. Wartości 8,33 oznaczają, że dany sonar nie widzi przeszkody;
- *SeeRobot* - informacja że robot widzi innego robota. Dany jest numer robota oraz kąt i odległość. Tutaj robot R1 nie widzi innego robota;
- *SeeLandmark* - informacja że robot widzi znacznik. Komunikat zawiera informację o treści znacznika (np. „r1"), odległości i kącie, oraz o koszcie obsługi związanym ze znacznikiem;
- *Hear* - Informacja o wiadomościach odebranych od innych robotów (zwykły tekst);
- *Bumper* - informacja z czujników dotykowych robota;
- *Position* - Informacja o pozycji robota we współrzędnych lokalnych danego pomieszczenia;



Rysunek 5.2: Klient - robot nr 1 z rys. 5.1. Robot widzi ścianę i znacznik „r1";

- *Global Pos* - Informacja o pozycji robota we współrzędnych komputera (lewy górny róg ekranu jest punktem (0,0). *Position* i *Global Pos* są wykorzystywane gdy symulowany przypadek zakłada, że robot ma znać swoją pozycję;

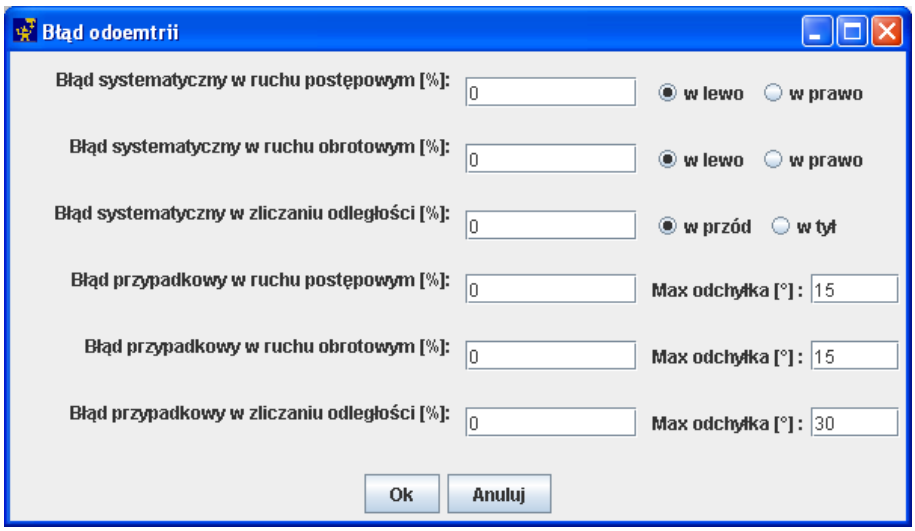
Komendy wykorzystane w komunikatach przesyłanych od klienta do serwera są następujące:

- *dash* - jazda prostoliniowa w przód lub w tył o określoną liczbę piksli;
- *turn* - obrót o określony kąt;
- *say* - wysłanie komunikatu;

5.6 Ważniejsze opcje symulatora

5.6.1 Odometria

Dla każdego robota istnieje możliwość ustawienia systematycznych i przypadkowych błędów odometrii. Okno z takimi ustawieniami przedstawione jest na rys. 5.3. W toku prac związanych z niniejszą pracą opcja ta nie była wykorzystywana.



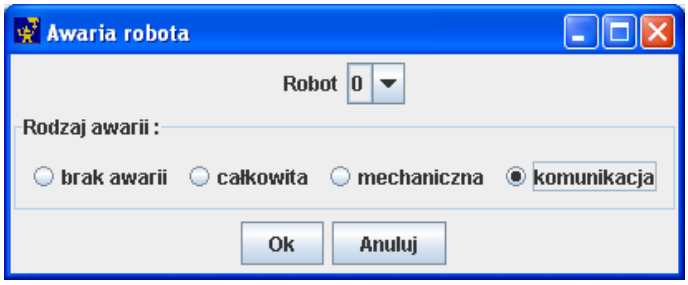
Rysunek 5.3: Ustawianie systematycznych i przypadkowych błędów odometrii robota. Domyślnie błędy nie są ustawione;

5.6.2 Awaria robota

Istnieje możliwość ustawienia awarii robota. Awaria może być mechanicza, komunikacyjna lub całkowita. Awaria mechaniczna oznacza, że robot może się komunikować, ale nie może się poruszać i powinien być traktowany przez inne roboty jako przeszkoda. Awaria komunikacyjna oznacza, że robot nie może komunikować się z innymi robotami ale może się przemieszczać. W przyprowadzanych eksperymentach rozpatrywano całkowitą (czyli mechaniczną i komunikacyjną) awarię robota. Awarię wybranego robota ustawia się w serwerze wybierając w menu Opcje-Awaria robota. Okno 5.4 przedstawia możliwość wyboru awarii robota.

5.6.3 Zmiana położenia robota

Przed rozpoczęciem eksperymentu oraz w jego trakcie istnieje możliwość ręcznej zmiany położenia robota. W celu umieszczenia wybranego robota w nowym położeniu należy podać nowe współrzędne robota lub wskazanie nowego położenia robota za pomocą myszki.



Rysunek 5.4: Ustawianie awarii robota w trakcie wykonywania doświadczenia;

Okno do zmiany położenia robota znajduje się w serwerze w menu *Opcje-Zmiana położenia robota*. Okno jest przedstawione na rys.5.5

Rysunek 5.5: Zmiana położenia robota w trakcie wykonywania doświadczenia;

Rozdział 6

Zakończenie

W pracy zastosowano mechanizm współpracy robotów w zadaniu poszukiwania obiektów w zadanym obszarze, ze znaną mapą. Zastosowano hierarchiczny podział mapy. Przeszukiwany obszar (budynek) został podzielony na podobszary (pomieszczenia). Taki sposób przedstawienia mapy umożliwia analizę mapy w dwojaki sposób:

- lokalnie - ograniczając się tylko do jednego pomieszczenia;
- globalnie - traktując drzwi pomiędzy pomieszczeniami jako węzły grafu;

Do poruszania się w obrębie każdego pomieszczenia (lokalnie) wykorzystano metodę A*. Globalne planowanie ścieżki odbywa się na podstawie analizy grafu, w którym węzłami są drzwi pomiędzy pomieszczeniami. Roboty nie posiadają lidera, ale są ponumerowanie i zawsze robot z niższym numerem ma pierwszeństwo decyzji przed robotem z wyższym numerem. Jest to rozwiązanie pośrednie pomiędzy grupą robotów równorzędnych, które muszą negocjować sposób rozwiązania zadania, a grupą z wyróżnionym liderem, który decyduje o akcji podejmowanej przez każdego robota. W toku badań przeprowadzono eksperymenty dla trzech różnych rozkładów pomieszczeń, z różnym rozkładem poszukiwanych obiektów (skupione i rozrzucone). W prezentowanym modelu negocjaji wprowadzono pojęcie *współczynnika chęci współpracy*, który określa wolę robotów do porzucenia wcześniej zaplanowanego zadania i pomocy robotowi, który rozpoczął aukcję.

6.1 Wnioski

1. Elementem nowym jest wykorzystanie mechanizmu aukcji przy znalezieniu poszukiwanego obiektu przez robota. Aukcja jest wykorzystana do znalezienia optymalnej liczby robotów do szybszego przeszukania danego pomieszczenia;
2. Zespół robotów wykonuje zadanie stabilnie, to znaczy, strata jednego lub kilku robotów nie powoduje przerwania wykonania zadania;

3. Prezentowany model jest elastyczny, co oznacza, że ewentualna rozbudowa i dodanie nowych funkcjonalności nie powinna wpływać na już istniejące elementy;

4. W wyniku przeprowadzonych eksperymentów pokazano, że w wypadku gdy poszukiwane obiekty są skupione i występują w małej ilości pomieszczeń, mechanizm negocjacji przyspiesza znalezienie wszystkich obiektów o 25-50%, w zależności od rodzaju mapy. W wypadku rozrzucenia obiektów w obszarze całej mapy, mechanizm negocjacji nie wpływa lub nieznacznie spowalnia wykonanie zadania;

5. W wyniku przeprowadzonych eksperymentów pokazano, że dla danego rozkładu mapy, ze skupionymi poszukiwanymi obiektami i dla danej liczby robotów, można tak dobrać współczynnik dojazdu, żeby mechanizm aukcji przyspieszał wykonanie zadania, co stanowi rozwiązanie postawionego zadania;

6.2 Kierunki dalszych badań

W trakcie pracy nad prezentowanym modelem pojawiały się nowe kierunki badań, które w toku niniejszych prac nie zostały rozwinięte a byłyby dobrym uzupełnieniem i rozbudową modelu. Należą do nich:

1. Analityczny dobór *współczynnika chęci dojazu* μ uzależniony od mapy i ilości robotów w zespole;

2. Rozbudowanie akcji robota po awarii mechanicznej lub komunikacyjnej. W pracy pokazano, że po awarii robota system wykona zadanie do końca, ale nie będzie to już zachowanie optymalne;

3. Wprowadzenie niepewności położenia robota. W chwili obecnej robot na skutek błędów numerycznych może zboczyć z zaplanowajej trasy, ale w dowolnej chwili może dostać informację o swoim rzeczywistym położeniu;

4. Wprowadzenie niepełnej wiedzy o mapie otoczenia. Na przykład przez wprowadzenie nieznanych przeszkód. Wydaje się, że stosunkowo niewielka rozbudowa modelu (modyfikacja grafu powstałego w wyniku rozszerzenia ścian i przeszkód w pomieszczeniu) umożliwi działanie w takich warunkach;

5. Wprowadzenie heterogeniczności w zespole;

6. Wprowadzenie elementu ludzkiego w zespole. Ciekawym zadaniem byłoby zbadanie, czy jeżeli jednym z członkow zespołu będzie człowiek, zadanie będzie wykonywane bardziej efektywnie;

7. Rozwiązywanie trudniejszych zadań, na przykład poszukiwanie zbiega, czyli możliwość przemieszczania się poszukianych obiektów;

Bibliografia

[1] P. Agre and D. Chapman. PENGI: An implementation of a theory of activity. In P Maes, editor, *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 268–272, Seattle, 1987. The MIT Press.

[2] P. E. Agre and S. J. Rosenschein. *Computational Theories of Interaction and Agency*. The MIT Press, Cambridge, 1996.

[3] K.S. Ali and R.C. Arkin. Multiagent teleautonomous behavioral control. *International Conference on Simulation of Adaptive Behavior*, 3:473–478, 1994.

[4] Stanislaw Ambroszkiewicz. entish: An approach to service description and composition.

[5] Stanislaw Ambroszkiewicz, Olaf Matyja, and Wojciech Penczek. Team formation by self-interested mobile agents. In *Selected Papers from the 4th Australian Workshop on Distributed Artificial Intelligence, Multi-Agent Systems*, pages 1–15, London, UK, 1998. Springer-Verlag.

[6] Pedro Aparício and Pedro Lima. Autonomous distributed control of a population of cooperative robots. In *Proc. of SIRS 99*, 1999.

[7] R. Arkin and K. Ali. Integration of reactive and telerobotic control in multi-agent robotic systems, 1994.

[8] R. Arkin and T. Balch. Cooperative multiagent robotic systems, 1998.

[9] R.C. Arkin. *Behavior-Based Robotics*. The MIT Press, Cambridge, 1998.

[10] M. Baglietto, M. Paolucci, L. Scardovi, and R. Zoppoli. Information-based multiagent exploration. In *Proceedings of the Third International Workshop on Robot Motion and Control, 2002.*, pages 173–179. RoMoCo'02, 2002.

[11] T. Balch and R. Arkin. Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.

[12] T. Balch and M. Hybinette. Social potentials for scalable multirobot formations, 2000.

[13] Howard Barringer, Michael Fisher, Dov Gabbay, Graham Gough, and Richard Owens. MetateM: A framework for programming in temporal logic. Technical Report UMCS-89-10-4, REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness (LNCS), 1989.

[14] Andreas Birk, Silvia Coradeschi, and Satoshi Tadokoro, editors. *RoboCup 2001: Robot Soccer World Cup V*, volume 2377 of *Lecture Notes in Computer Science*. Springer, 2002.

[15] A. Borkowski and M. Gnatowski. Krzepki model współpracy zespołu jednorodnych robotów w zadaniu przeszukiwania pomieszczenia. In *Automation '02*, Warszawa, 2002. Konferencja Naukowo-Techniczna: Automatyzacja - Nowości i perspektywy.

[16] A. Borkowski, M. Gnatowski, and J. Malec. Mobile robot cooperation in simple environments. In *Proc. of the 2nd IEEE Int.*, pages 109–114, Bukowy Dworek, October 2001. Workshop on Robot Motion and Control.

[17] M. E. Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, Cambridge, 1987.

[18] Michael E. Bratman, David Israel, and Martha Pollack. Plans and resource-bounded practical reasoning. In Robert Cummins and John L. Pollock, editors, *Philosophy and AI: Essays at the Interface*, pages 1–22. The MIT Press, Cambridge, Massachusetts, 1991.

[19] Rodney A Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, March 1986.

[20] S. Bussmann and J. Mueller. A communication architecture for cooperating agents. *Computers and Artificial Intelligence*, 12:37–53, 1993.

[21] Nicola Ceccarelli, Mauro Di Marco, Andrea Garulli, and Antonio Giannitrapani. A decentralized control law for collective circular motion of a team of nonholonomic mobile robots. 2004.

[22] Krzysztof Cetnarowicz. M-agent architecture based method of development of multiagent systems. In *The 8th Joint EPS-APS International Conference on Physics Computing, ACC CYFRONET*, Kraków, Poland, 1996.

[23] Krzysztof Cetnarowicz, Pablo Gruer, Vincent Hilaire, and Abder Koukam. A Formal Specification of M-Agent Architecture. In *CEEMAS '01: Revised Papers from the Second International Workshop of Central and Eastern Europe on Multi-Agent Systems*, pages 62–72, London, UK, 2002. Springer-Verlag.

[24] A. Chmielniak, A. Dubrawski, and B.Siemiątkowska. Controlling teams of mobile robots. In *Proc. of International Conference on Intelligent Techniques in Robotics*, Warszawa, 1999. Control and Decision Making, Polish-Japanise Institute of Information Technology.

[25] A. Chmielniak, A. Dubrawski, and B.Siemiątkowska. A distributed system for control and management of teams of mobile robots. In *Automation '99*, Warszawa, 1999. Konferencja Naukowo-Techniczna: Automatyzacja - Nowości i perspektywy.

[26] Andrzej Chmielniak. *Metoda planowania ścieżek robotów mobilnych prowadzących inspekcję*. Wydział MEiL, Politechnika Warszawska, 2003. Rozprawa doktorska.

[27] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.

[28] Jung D. and Zelinksy A. An architecture for distributed cooperative planning in a behaviour-based multi-robot system. In *Robotics and Autonomous Systems*, volume 26 (2), pages 149–174. Elsevier Science, 1999.

[29] Remco de Boer, Jelle Kok, and Frans Groen. Multi-agent systems: The uva trilearn 2001 robotic soccer simulation team.

[30] Mark d'Inverno, David Kinny, Michael Luck, and Michael Wooldridge. A formal specification of dMARS. In *Agent Theories, Architectures, and Languages*, pages 155–176. Springer-Verlag, 1997.

[31] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multi-agent robotics, 1996.

[32] Barbara Dunin-Keplicz and Rineke Verbrugge. Evolution of collective commitment during teamwork. *Fundamenta Informaticae*, 56, 2003.

[33] P. Dunne, M. Wooldridge, and M. Laurence. The complexity of contract negotiation, 2003.

[34] Edmund Durfee, Victor Lesser, and Daniel Corkill. Coherent Cooperation Among Communicating Problem Solvers. *IEEE Transactions on Computers*, C36(11):1275–1291, November 1987.

[35] Oren Etzioni. Intelligence without robots: A reply to brooks. *AI Magazine*, 14(4):7–13, 1993.

[36] Jacques Ferber. Reactive distributed artificial intelligence: principles and applications. *Foundations of distributed artificial intelligence*, 1996.

[37] Jacques Ferber. *Multi-Agent Systems*. Harlow: Addison Wesley Longman, 1999.

[38] T. Finin, D. McKay, and R. Fritzson. An Overview of KQML: A Knowledge Query and Manipulation Language. Technical report, University of Maryland Computer Science Department, 1992.

[39] J. A. Firby. An investigation into reactive planning in complex domains. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 202–206, Milan, Italy, 1987.

[40] K. Fischer, J. Müller, and M. Pischel. A pragmatic BDI architecture. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intellingent Agents II (LNAI)*, volume 1037, pages 203–218, Berlin, Germany, 1996. Springer-Verlag.

[41] M. Fisher. A survey of concurrent METATEM – the language and its applications. In D. M. Gabbay and H. J. Ohlbach, editors, *Temporal Logic - Proceedings of the First Intemational Conference (LNAI Volume 827)*, pages 480–505. Springer-Verlag: Heidelberg, Germany, 1994.

[42] S. Franklin and A. Graesser. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In *Intelligent Agents III. Agent Theories, Architectures and Languages (ATAL'96)*, volume 1193, Berlin, Germany, 1996. Springer-Verlag.

[43] L. Gasser and J. P. Briot. Object-based concurrent programming and DAI. In *Distributed Artificial Intelligence: Theory and Praxis*, pages 81–108, Boston, 1992. Kluwer Academic Publishers.

[44] M. R. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence*. San Mateo, 1987.

[45] M. P. Georgeff and A. S. Rao. A profile of the Australian AI Institute. *IEEE Expert*, 11(6):89–92, December 1996.

[46] Michael P. Georgeff and Amy L. Lansky. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 677–682, Seattle, 1987. The MIT Press.

[47] R. Ghanea-Hercock and D. P. Barnes. Coupled behaviors in the reactive control of cooperation mobile robots. *Advanced Robotics*, 10:161–177, 1996.

[48] M. Gnatowski and J. Malec. Minimalistyczny model współpracy robotów mobilnych. In *VII Krajowa Konferencja Robotyki*, pages 205–216, Wrocław, 2000. Oficyna Wydawnicza Politechniki Wrocławskiej.

[49] D.D. Grossman. Traffic control of multiple robot vehicles. *Robotics and Automation*, 4:491–497, 1988.

[50] Afsaneh Haddadi. Towards a pragmatic theory of interactions. In *Proc. International Conference on MultiAgent Systems ICMAS*, pages 133–139, San Francisco, 1995.

[51] Afsaneh Haddadi. *Communication and cooperation in agent systems: A pragmatic theory*, volume 1056 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1996.

[52] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat. An overview of expert systems. In F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, editors, *Building Expert Systems*, pages 3–29. Addison-Wesley, London, 1983.

[53] Rufus Isaacs. *Differetial Games*. John Wiley and Sons, New York, 1965.

[54] N. R. Jennings, J. M. Corera, I. Laresgoiti, E. H. Mamdani, F. Perriollat, P. Skarek, and L. Z. Varga. Using ARCHON to develop real-world DAI applications for electricity transportation management and particle accelerator control. *IEEE Expert*, 11(6):60–88, 1996.

[55] L. P. Kaelbling. An architecture for intelligent reactive systems. In M. P. Georgeff and A. L. Lansky, editors, *Reasoning about Actions and Plans - Proceedings of the 1986 Workshop*, pages 395–410, San Mateo, 1986. Morgan Kaufmann Publishers.

[56] Leslie P. Kaelbling and Stanley J. Rosenschein. Action and planning in embedded agents. In P Maes, editor, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, pages 35–48, Cambridge, 1990. The MIT Press.

[57] Leslie Pack Kaelbling. A situated-automata approach to the design of embedded agents. *SIGART Bull.*, 2(4):85–88, 1991.

[58] Gal A. Kaminka, Pedro U. Lima, and Raúl Rojas, editors. *RoboCup 2002: Robot Soccer World Cup VI*, volume 2752 of *Lecture Notes in Computer Science*. Springer, 2003.

[59] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. RoboCup: The robot world cup initiative. In W. Lewis Johnson and Barbara Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 340–347, New York, 5–8, 1997. ACM Press.

[60] J. Kok, M. Spaan, and N. Vlassis. Multi-robot decision making using coordination graphs.

[61] Brian Komar. *TCP/IP dla każdego*. Helion, Gliwice, 2002.

[62] K. Konolige. *A Deduction Model of Belief*. Pitman Publishing, London and Morgan Kaufmann, San Mateo, 1986.

[63] K. Kosuge, T. Oosumi, H. Asama, T.Fujii, and H.Kaestsu. Coordinate motion control of multiple autonomous mobile robots based on compliant motion control. In *Distributed Autonomous Robotic Systems 3*, pages 141–150, Berlin, 1998. Springer-Verlag New York, LLC.

[64] Tim Laue and Thomas Röfer. A behavior architecture for autonomous mobile robots based on potential fields. In *RoboCup 2004*, LNAI. springer, 2004.

[65] Yves Lesperance, Hector J. Levesque, Fangzhen Lin, Daniel Marcu, Raymond Reiter, and Richard B. Scherl. Foundations of a logical approach to agent programming. In *Agent Theories, Architectures, and Languages*, pages 331–346, 1995.

[66] Józef Lisowski. Wykorzystanie teorii gier do podejmowania decyzji manewrowej w nawigacji morskiej. *Biuletyn Automation 2003*, pages 39–63, 2003.

[67] P. Maes. *Designing Autonomous Agents*. The MIT Press, Cambridge, 1990.

[68] Patti Maes. Situated agents can have goals. In Patti Maes, editor, *Designing Autonomous Agents*, pages 49–70. MIT Press, 1990.

[69] Pattie Maes. The dynamics of action selection. In *Proceedings of the Eleventh International Joint Conference on Articial Intelligence (IJCAI89)*, volume 2, pages 991–997, 1989.

[70] Pattie Maes. The agent network architecture (ANA). *SIGART Bull.*, 2(4):115–120, 1991.

[71] http://www.informatik.uni-bonn.de/ rhino/,
http://www.cs.cmu.edu/ minerva/about.html.

[72] J. P. MMüller. A cooperation model for autonomous agents. In *ECAI '96: Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*, pages 245–260, London, UK, 1997. Springer-Verlag.

[73] Stewart Moorehead, Reid Simmons, and William Red L. Whittaker. Autonomous exploration using multiple sources of information. In *IEEE International Conference on Robotics and Automation*, May 2001.

[74] J. Müller. A cooperation model for autonomous agents. In M. Wooldridge, J. P. Müller, and N. R. Jennings, editors, *Intellingent Agents III (LNAI)*, volume 1139, pages 245–260, Berlin, Germany, 1997. Springer-Verlag.

[75] J. Müller, M. Pischel, and M. Thiel. Modeling reactive behaviour in vertically layered agent architectures. In M. Wooldridge and N. R. Jennings, editors, *Intellingent Agents: Theories, Architectures and Languages (LNAI)*, volume 890, pages 261–276, Berlin, Germany, January 1995. Springer-Verlag.

[76] J. P. Muller, M. Wooldridge, and N. R. Jennings, editors. *Intelligent Agents III. Agent Theories, Architectures, and Languages ECAI'96 Workshop (ATAL)*, Lecture Notes in Artificial Intelligence, Vol. 1193, Berlin, 1996. Springer-Verlag.

[77] Daniele Nardi, Martin Riedmiller, Claude Sammut, and José Santos-Victor, editors. *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Computer Science*. Springer, 2005.

[78] N. J. Nilsson. Towards agent programs with circuit semantics. Technical Report STAN–CS–92–1412, Computer Science Department, Stanford, CA 94305, 1992.

[79] Esben H. Ostergaard, Gaurav S. Sukhatme, and Maja J. Matari. Emergent bucket brigading: a simple mechanisms for improving performance in multi-robot constrained-space foraging tasks. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 29–30, New York, NY, USA, 2001. ACM Press.

[80] L. Parker. Cooperative motion control for multi-target observation, 1997.

[81] http://www.robothalloffame.org/mars.html,
http://mars.jpl.nasa.gov/mpf/rover/sojourner.html.

[82] Daniel Polani, Brett Browning, Andrea Bonarini, and Kazuo Yoshida, editors. *RoboCup 2003: Robot Soccer World Cup VII*, volume 3020 of *Lecture Notes in Computer Science*. Springer, 2004.

[83] Janusz Racz. *Elementy sztucznej inteligencji w systemach nawigacyjnych autonomicznych mobilnych robotów*. IPPT PAN, Warszawa, 1995. Rozprawa doktorska.

[84] A. S. Rao. Decision procedures for propositional linear-time Belief-Desire-Intention logics. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents II (LNAI)*, volume 1037, pages 33–48, Berlin, 1996. Springer-Verlag.

[85] Anand S. Rao and M. P. Georgeff. A model-theoretic approach to the verification of situated reasoning systems. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 318–324, Chamberry, France, 1993.

[86] Anand S. Rao and Michael P. Georgeff. Asymmetry thesis and side-effect problems in linear-time and branching-time intention logics. In John Myopoulos and Ray Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial*

Intelligence (IJCAI-91), pages 498–505, Sydney, Australia, 1991. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[87] Anand S. Rao and Michael P. Georgeff. Modeling rational agents within a BDI-architecture. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1991.

[88] Anand S. Rao and Michael P. Georgeff. An abstract architecture for rational agents. In W. Swartout C. Rich and B. Nebel, editors, *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 439–449. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1992.

[89] J. Reif and H. Wang. Social potential fields: A distributed behavioral control for autonoomous robots. In K. Goldberg, D. Halperin, J.-C. Latombe and R. Wilson, editors, *Robotics and Autonomous Systems*, volume 27 (3), pages 171–194. International Workshop on Algorithmic Foundations of Robotics (WAFR), 1999.

[90] http://www.robocup.org/overview/21.html.

[91] http://www.rescuesystem.org/robocuprescue/.

[92] Jeffrey S. Rosenschein and Gilad Zlotkin. Designing Conventions for Automated Negotiation. *AI Magazine*, pages 29–46, 1994.

[93] Jeffrey S. Rosenschein and Gilad Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers.* The MIT Press, Cambridge, July 1994.

[94] S Rosenschein and L Kaelbling. The synthesis of digital machines with provable epistemic properties. In *Proceedings of the 1986 Conference on Theoretical aspects of reasoning about knowledge*, pages 83–98, San Francisco, CA, USA, 1986. Morgan Kaufmann Publishers Inc.

[95] S. J. Rosenschein and L. P. Kaelbling. A situated view of representation and control. In P. E. Agre and S. J. Rosenschein, editors, *Computational Theories of Interaction and Agency*, pages 515–540. The MIT Press, Cambridge, MA, 1996.

[96] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, 1995.

[97] M. J. Schoppers. Universal plans for reactive robots in unpredictable environments. In John McDermott, editor, *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 1039–1046, Milan, Italy, 1987. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[98] John R. Searle. *An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, 1969.

[99] Barbara Siemiątkowska. *Rastrowa reprezentacja otoczenia w sterowaniu ruchomym robotem*. IPPT PAN, Warszawa, 1996. Rozprawa doktorska.

[100] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes. Coordination for multi-robot exploration and mapping. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2000.

[101] M. P. Singh. A critical examination of the Cohen-Levesque theory of intention. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-92)*, pages 364–368, Vienna, Austria, 1992.

[102] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, 1981.

[103] L. Steels. Cooperation between distributed agents through self-organization. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI - Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-89)*, Elsevier Science Publishers B.V., pages 175–196, Amsterdam, The Netherlands, 1990.

[104] Gita Sukthankar. Team-aware multirobot strategy for cooperative path clearing. In *AAAI-2000*, July 2000.

[105] http://pl.wikipedia.org/wiki/system wieloagentowy.

[106] Piort Szynkarczyk. *Zastosowanie równoległej architektury warstwowej i problem wyboru akcji w sterowaniu systemów autonomicznych*. Wojskowa Akademia Techniczna, 1999. Rozprawa doktorska.

[107] Z.D. Wang, T. Takahashi, T. Nitsuma, T. Ninjouji, and E. Nakano. Logue: An architecture for task and behavior object transmission among multiple autonomous robots. In *Robotics and Autonomous Systems*, volume 44, pages 261–271. Elsevier Science, 2003.

[108] Gerhard Weiss. *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, Cambridge, Massachusetts London, England, 2000.

[109] M. Wooldridge. Agent-based software engineering. *IEE Proceedings Software Engineering*, 144(1):26–37, 1997.

[110] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engneering Review*, 10(2):115–152, 1995.

[111] X.Rui, C.Pingyuan, and X.Xiaofei. Realization of multi-agent planning system for autonomous spacecraft. *Advances in Engineering Software*, 36:266–272, 2005.

[112] Teresa Zielińska. Wykorzystanie własności chodu człowieka i zwierząt do syntezy ruchu maszyn kroczących. In *Prace Instytutu Biocybernetyki i Inżynierii Biomedycznej*, volume 40, Warszawa, 1995.

[113] Cezary Zieliński. Formal approach to the design of control software for assstive robots in advanced manufacturing systems. In *Seminar on Human-Rotob Collaboration in Manufacturing, SIMTech*, Singapore Institute of Manufacturing Technology, sierpień 2005.