ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2015
Seria: ORGANIZACJA I ZARZĄDZANIE z. 86 Nr kol. 1946

Witold BRANDYS
Wyższa Szkoła Technologii Informatycznych w Katowicach
witoldbrandys@wsti.edu.pl
Adam GAŁUSZKA
Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki
adam.galuszka@polsl.pl

ROZWIĄZYWANIE KONFLIKTU W ŚRODOWISKU WIELOAGENTOWYM Z WYKORZYSTANIEM ELEMENTÓW HIERARCHICZNEJ TEORII GIER

Streszczenie. W artykule zaprezentowane zostało środowisko wieloagentowe, w którym każdy z agentów dąży do osiągnięcia własnego celu. Zakłada się, że preferencje osiągnięcia poszczególnych podcelów, składających się na cel, mogą być różne dla poszczególnych agentów, natomiast same cele mogą być sprzeczne ze sobą. Dodatkowo zakłada się strukturę hierarchiczną wśród agentów, wynikającą z różnych możliwości ich oddziaływania na otoczenie.

Słowa kluczowe: planowanie, sztuczna inteligencja robotów, gry hierarchiczne, gry niekooperacyjne, równowaga Stackelberga.

HIERARCHICAL GAME APPROACH TO SOLVE CONFLICTS IN MULTIAGENT SYSTEMS

Summary. The article presents multi-agent environment, in which each agent seeks to achieve its objective. It is assumed that the preferences of achieving the specific sub-goals which define the goal may be different for individual agents, and agent goals may be in conflict with each other. Additionally it is assumed a hierarchical structure among agents, due to different possibilities of their impact on the environment.

Keywords: planning, artificial intelligence of robots, hierarchical games, noncooperative games, Stackelberg equilibrium.

1. Wprowadzenie

W środowisku, w którym znajduje się wiele agentów działających niezależnie, przy czym każdy z nich ma niepełną wiedzę na temat otoczenia, starając się zrealizować swoje własne cele, powstają konflikty interesów. Porozumienie między agentami może być osiągnięte

przez wymianę informacji i negocjację. W takim przypadku każdy z działających agentów może starać się: przedstawić argumenty przekonujące do wykonania jego celu, zaoferować w zamian istotne informacje albo wykonanie jakiejś akcji, której inny agent nie jest w stanie wykonać [1]. W celu uzyskania powyższych efektów niezbędna staje się wymiana informacji między agentami. Współcześnie spotykane w literaturze rozwiązania często prowadzą do podejść hierarchicznych, w których wybrane agenty pełnią rolę tzw. leaderów, a inne – tzw. followerów (np. [11], [12], [13]).

Istnieją jednak sytuacje, w których komunikacja jest niemożliwa. Może to być spowodowane na przykład przez: znalezienie się w środowisku uniemożliwiającym kontakt, wysoki koszt utrzymywania łączności lub awarię systemu komunikacji. Istnieją sytuacje, w których nie ma możliwości kontaktu albo też nie można czekać na jego przywrócenie. W takim przypadku należy w systemie zastosować algorytm, który pozwoli agentom podejmować decyzję samodzielnie. Szczególnym przypadkiem takiej sytuacji jest taka, w której komunikacja jest jednokierunkowa. Oznacza to, że jeden z agentów nie ma żadnej informacji o działaniach pozostałych agentów, natomiast oni mogą reagować mając dostęp do informacji im przekazywanej. Z podobną sytuacją mamy do czynienia, gdy jeden z agentów musi podjąć działanie, nie czekając na akcje pozostałych.

Zaproponowana w artykule metoda znalezienia rozwiązania wykorzystuje własności odwracalności niektórych systemów planowania w sztucznej inteligencji oraz metodologię poszukiwania rozwiązania w warunkach niejednoznacznej informacji o sytuacji początkowej problemu. Ponieważ znalezione rozwiązanie w ogólnym przypadku jest niekompletne (niedokładnie precyzuje akcje agentów), więc pokazano w jaki sposób sprecyzować plan wykorzystując elementy hierarchicznej teorii gier (równowagi niekooperacyjnej Stackelberga).

Problemy planowania tego typu mogą być modelowane jako system STRIPS (w środowisku Świata Klocków), z jednym stanem początkowym i wieloma, alternatywnymi stanami docelowymi.

Jeśli problem planowania STRIPS jest odwracalny, to możliwe jest zastosowanie mechanizmu planowania w obecności niekompletnej informacji do rozwiązania problemu odwrotnego i znalezienia rozwiązania problemu oryginalnego [2].

Ilustracją sytuacji z wieloma agentami może być przykład, w którym rolę agentów pełnią roboty z chwytakami, zdolnymi do przemieszczania obiektów w środowisku Świata Klocków.

Przedmiotem pracy jest znalezienie rozwiązania konfliktów pomiędzy agentami oraz zbadanie możliwości zastosowania teorii gier. Ze względu na istniejącą hierarchię, do sprecyzowania planu wykorzystano równowagę niekooperacyjną von Stackelberga.

1.1. Definicja problemu – założenia

W celu modelowania badanej sytuacji przyjęto następujące założenia [2]:

- Roboty mają autonomię działań – pełnią rolę agentów.
- Stan początkowy zawiera skończoną liczbę klocków na stole o nieograniczonej ilości miejsca.
- Dwa (lub w ogólnym przypadku więcej) roboty (agenty) próbują przekształcić stan początkowy, każdy z nich w odrębny sposób (każdy agent chce osiągnąć swój własny cel).
- Cel każdego z robotów składa się z podcelów.
- Każdy podcel ma swoje preferencje (podcele są mniej lub bardziej istotne dla agentów) wyrażone wartościami liczbowymi.
- Agenty mają odmienne możliwości oddziaływania na środowisko zewnętrzne. Dany robot nie musi być w stanie wykonywać wszystkich operacji, prowadzących do realizacji jego podcelów.
- Agenty nie mogą współpracować (to założenie jest uzasadnione np. w przypadku środowiska, w którym komunikacja jest zabroniona lub urządzenia służące do komunikacji ulegają awarii).
- Jeden z agentów musi wykonać akcję nie czekając na pozostałych. Taka sytuacja może mieć miejsce, gdy jeden z robotów pracuje w systemie czasu rzeczywistego i kluczowe dla niego jest ukończenie zadania w określonym czasie. Zadaniem drugiego agenta jest racjonalna reakcja na decyzję agenta pierwszego.
- Zysk rozumiany jest jako suma wartości zrealizowanych podcelów.
- Rozwiązanie będzie rozumiane jako znalezienie planu (sekwencji akcji) prowadzącego do maksymalizacji zysków każdego z agentów.

1.2. System STRIPS

System STRIPS (STanford Research Institute Problem Solver) do modelowania otoczenia robota wykorzystuje klasyczne środowisko zwane Światem Klocków (Blocks World) [4].

Dziedzinę Świata Klocków można opisać za pomocą systemu STRIPS przez cztery listy (C; O; I; G) [5]:

• C – skończony zestaw podstawowych formuł/atomów (faktów), zwanych warunkami (ang. „Conditions”).
• O – skończony zestaw operatorów (ang. „Operators”), zwany również akcjami.
• I – skończony zestaw predykatów wskazujących stan początkowy (ang. „Initial state”).
• G – skończony zestaw predykatów wskazujących stan docelowy (ang. „Goal state”).

Dla przykładowej sytuacji świata klocków zbiorem faktów może być lista:

(on x y) – klocek x leży na klocku y,

(on-table x) – klocek x leży na stole,

(clear x) – na klocku x nie leży żaden klocek,

(arm-empty) – ramię robota jest puste,

(holding x) – robot trzyma klocek x.

Operatory (O) w reprezentacji STRIPS mają:

- nazwę, która może zawierać argumenty – na przykład PICK-UP x, gdzie *x* jest argumentem, a *PICK-UP* nazwą,
- listę warunków wstępnych (pre) (ang. „precondition list"), czyli listę faktów, które muszą być prawdziwe, aby można było zastosować akcję,
- listę skreśleń (del) (ang. „delete list"), czyli listę faktów, które przestają być prawdziwe po zastosowaniu operatora,
- listę dopisków (add) (ang. „add list"), czyli listę faktów, które stają się prawdziwe po zastosowaniu operatora.

Budowa operatora (PICK-UP x), oznaczającego podniesienie klocka ze stołu, wygląda następująco [3]:

Lista warunków wstępnych (pre): (on-table x), (clear x), (arm-empty).

Lista skreśleń (del): (on-table x), (clear x), (arm-empty).

Lista dopisków (add): (holding x).

Operatorów zaś:

(PUT-DOWN x) – położenie klocka na stół.

Lista warunków wstępnych (pre): (holding x).

Lista skreśleń (del): (holding x).

Lista dopisków (add): (on-table x), (clear x), (arm-empty).

(STACK x y) – położenie klocka na stos (inny klocek).

Lista warunków wstępnych (pre): (holding x), (clear y).

Lista skreśleń (del): (holding x), (clear y).

Lista dopisków (add): (arm-empty), (on x y), (clear x).

(UNSTACK x y) – podniesienie klocka ze stosu (z innego klocka).

Lista warunków wstępnych (pre): (arm-empty), (clear x), (on x y).

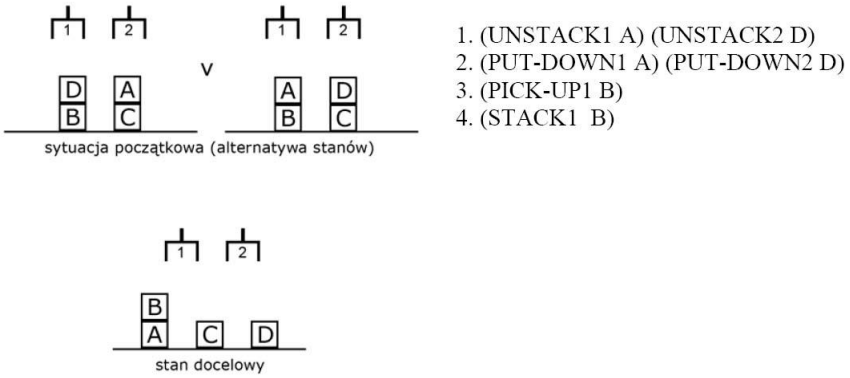
Lista skreśleń (del): (arm-empty), (clear x), (on x y).

Lista dopisków (add): (holding x), (clear y).

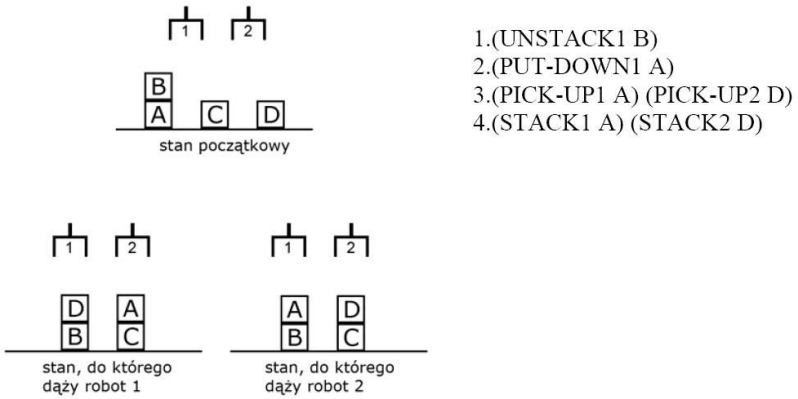
Rozwiązaniem problemu typu STRIPS jest znalezienie zbioru operatorów przekształcających stan początkowy w stan docelowy świata z zadania.

1.3. System STRIPS dla problemu planowania w środowisku wielu agentów jako problem odwrotny do świata klocków z niejednoznaczną sytuacją początkową

Problem, w którym jest wiele możliwych stanów początkowych (nie jesteśmy w stanie ustalić, który z nich naprawdę występuje) i jeden stan docelowy nazywany jest problemem planowania z niekompletną (niepełną) informacją [7]. W przypadku braku pełnej informacji o środowisku zewnętrznym, można poszukać planu odpornego. W takim przypadku analizujemy skutki zastosowania operatora w każdej możliwej sytuacji tak, aby osiągnąć stan docelowy, niezależnie od możliwego stanu początkowego. Niekompletny plan odporny to taki, w którym nie określono kompletu zmiennych dla operatora. Na przykład STACK1 B będzie oznaczać położenie klocka B przez robota 1 na jakimś innym – nieokreślonym (rys. 1).



Rys. 1. Niejednoznaczna sytuacja początkowa i plan odporny
Fig. 1. Ambiguous initial state and conformant plan



Rys. 2. Problem odwrotny z niekompletnym planem odpornym
Fig. 2. Inverse problem with incomplete conformant plan

Jako problem odwrotny do niego rozumiemy sytuację, w której jest jeden stan początkowy, a wiele stanów docelowych [8]. Odpowiada to problemowi planowania w środowisku wielu agentów (robotów), gdzie każdy z nich chce uzyskać swój własny cel (rys. 2). Źródłem konfliktu jest określenie wzajemnie sprzecznych podcelów poszczególnych agentów. Agent 1 (robot 1) dąży do ustawienia klocka D na klocku B, podczas gdy agent 2 (robot 2) dąży do ustawienia klocka D na klocku C. Podobnie z klockiem A. Jeżeli dodatkowo założymy, że manipulację klockami A i B może wykonywać agent 1, a klockami

C i D agent 2, to roboty muszą współpracować ze sobą, mimo sprzecznych celów końcowych. Podany na rys. 2 niekompletny plan odporny prowadzi do alternatywy stanów. Jeśli jesteśmy w stanie znaleźć plan dla problemu planowania w obecności niekompletnej informacji, to możliwe jest uzyskanie na jego podstawie rozwiązania dla planowania w środowisku wielu agentów [2]. W praktyce wiąże się to z zamianą operatorów na odwrotne (UNSTACK na STACK, PICK-UP na PUT-DOWN, STACK na UNSTACK i PUT-DOWN na PICK-UP) oraz odwróceniem kolejności operatorów.

Plany rozwiązujące problemy z rys. 1 i 2 są nieprecyzyjne, a tym samym niejednoznaczne. Jest to związane z tym, że podcele agentów (robotów) są ze sobą sprzeczne. Konfliktu nie da się uniknąć, jeśli poszukujemy jednego planu, biorącego pod uwagę różne sytuacje początkowe. Do jego rozwiązania zastosujemy techniki znane z teorii gier.

2. Rozwiązanie konfliktu z wykorzystaniem równowagi niekooperacyjnej

Na podstawie nieprecyzyjnego planu oraz równowagi niekooperacyjnej zostanie pokazane, jak otrzymać plan jednoznaczny, będący kompromisem dla sytuacji konfliktowej. Podstawą konstruowania gry, będącej ilustracją konfliktu pomiędzy robotami, są priorytety osiągania poszczególnych podcelów. Zakłada się, że priorytety te mogą reprezentować zyski.

2.1. Równowaga niekooperacyjna w strategiach czystych

Dla problemu gry pomiędzy dwoma graczami i reprezentacji macierzowej zakładamy, że gracz D_1 wybiera wiersze, a gracz D_2 kolumny macierzy. Obaj gracze starają się minimalizować wskaźnik jakości, zdefiniowany odpowiednio macierzami $MA = \{a_{ij}\}$ oraz $MB = \{b_{ij}\}$. Rezultatem wyboru pary strategii (i,j) jest zatem para wyników (a_{ij}, b_{ij}).

Przyjmijmy następujące założenie: agent 1 to robot pracujący w systemie czasu rzeczywistego, dla którego nadrzędnym priorytetem jest ukończenie zadania w określonym czasie.

Jeżeli założymy, że agent 2 może podążać za akcjami agenta 1, to sytuację konfliktową rozwiązujemy w sensie równowagi von Stackelberga. Role graczy są wtedy niesymetryczne – jeden z graczy (leader) ma możliwość forsowania swojej strategii w stosunku do drugiego gracza (followera). Zadaniem followera jest racjonalna reakcja na decyzje leadera.

Zbiór racjonalnych reakcji (optymalnych odpowiedzi) followera (gracz D2):

$R(i) = \{k : \forall_j\, b_{ik} \leq b_{ij}\}$

Strategie von Stackelberga dla leadera i_0 (S* – koszt dla leadera)

$\max_j \in R(i0)\, a_{i0\,j} = \min_i \max_j \in R(i)\, a_{ij} = S^*(A).$

Element $j \in R(i_0)$ to odpowiedź followera na strategię i_0 leadera. Para (i_0, j_0) jest rozwiązaniem równowagi von Stackelberga.

Z tego samego rozwiązania skorzystamy, gdy agent 2 wyprzedza akcje agenta 1. Zmieni się tylko ich rola. Leaderem zostanie agent 2, a followerem agent 1. Sposób rozwiązania sytuacji konfliktowej w sensie von Stackelberga, wraz z uzyskaniem kompletnego planu, pokazano w dalszej części artykułu wraz z podaniem przykładu.

Jeżeli agent 2 wykonuje działania w tym samym czasie co agent 1, to sytuację konfliktową rozwiązujemy w sensie równowagi Nasha (np. [6]), co pokazano w pracy [3].

2.2. Metoda rozwiązania problemu

Algorytm postępowania od wygenerowania problemu planowania w Świecie Klocków aż po uzyskanie kompletnego planu można streścić następująco.

Najpierw poszukujemy planu rozwiązującego problem z niejednoznaczną sytuacją początkową. Znaleziony plan zawiera „niekompletne" operatory STACK, jak na rys. 2. Zatem dany klocek może zostać położony przez robota (agenta) na dowolnym innym klocku, realizując przy tym albo swój podcel, albo podcel innego robota (agenta). Prowadzi to do sytuacji konfliktowej. Jeśli określimy różne priorytety podcelów, możliwe stanie się uzyskanie stanu prowadzącego do stanu docelowego, który będzie kompromisem dla wszystkich agentów (robotów), którzy znajdują się w konflikcie. Należy zauważyć, że zwykle kompromis taki wiąże się z osiągnięciem tylko niektórych podcelów dla poszczególnych agentów. Do otrzymania takiego kompromisu zaproponowano grę niekooperacyjną.

Przykład. Dziedzina problemu w języku PDDL jest zdefiniowana następująco:

```
(define (domain problem1) (:requirements :strips :equality :uncertainty)
(:predicates (on ?x ?y) (on-table ?x) (clear ?x) (arm1-empty) (arm2-empty)
(holding1 ?x) (holding2 ?x) )
(:action pick-up
        :parameters (?ob1)
        :precondition (and (clear ?ob1) (on-table ?ob1) (arm-empty))
  :effect (and (not (on-table ?ob1)) (not (clear ?ob1)) (not (arm-empty)) (holding ?ob1)))
(:action put-down
        :parameters (?ob)
        :precondition (holding ?ob)
        :effect (and (not (holding ?ob) (clear ?ob) (arm-empty) (on-table ?ob)))
(:action stack
        :parameters (?sob ?sunderob)
```

```
        :precondition (and (holding ?sob) (clear ?sunderob))
        :effect (and (not (holding ?sob)) (not (clear ?sunderob)) (clear ?sob) (arm-empty)
            (on ?sob ?sunderob)))
(:action unstack
        :parameters (?x)
        :precondition (and (clear ?x) (arm-empty))
        :effect (and (holding ?x) (not (clear ?x)) (not (arm-empty)) (when (on ?x C) (and
            (clear C) (not (on ?x C)) ) ) (when (on ?x D) (and (clear D) (not (on ?x D)).
```

Założenie, że robot 1 przemieszcza klocki A, B, C, a robot 2 przemieszcza klocki D, E, F spowodują modyfikację akcji stack na stack1 i stack2 oraz odpowiednio akcji unstack, pick-up i put-down na odpowiadające im akcje, uwzględniające manipulowanie wyłącznie wybranymi klockami.

Opis problemu planowania – sytuacje początkowa (*init*) oraz końcowa (*goal*) zostały zdefiniowane następująco (zapis w języku PDDL):

```
(define (problem s1) (:domain problem1) (:objects A B C D E F)
(:init (clear A) (clear E) (arm1-empty) (arm2-empty) (on-table B) (on-table F)
    (or (and (on C B) (on A C) (on D F) (on E D)
    (not (on A D)) (not (on E C)) (not (on D B)) (not (on C F)))
    (and (on A D) (on D B) (on E C) (on C F)
    (not (on A C)) (not (on C B)) (not (on E D)) (not (on D F)))          ))
(:goal (and (on A E) (on B D) (on C F) (on-table E) (on-table D) (on-table F))
```

Plan odporny, rozwiązujący problem, przybiera zatem następującą postać:

step7 – ((((stack2 E)))
step6 – (((pick-up2 E)) ((stack1 A)))
step5 – (((stack2 D)) ((unstack1 A)))
step4 – (((pick-up2 D)) ((stack1 C)))
step3 – (((put-down2 D)) ((unstack1 C)))
step2 – (((pick-up2 D)) ((put-down1 B)))
step1 – (((unstack1 B)))).

Powyższy plan jest nieprecyzyjny, np. operator stack1 A nie opisuje, na jakim klocku położyć klocek A. Wykonywanie odwrotnego planu odpornego przez obydwa roboty prowadzi do konfliktu. Ponieważ robot 1 wykonuje swoje akcje wcześniej niż robot 2, więc jego rozwiązanie można uzyskać przez zastosowanie równowagi niekooperacyjnej von Stackelberga. Niech macierz MA = $\{a_{ij}\}$ opisuje profit robota 1, a macierz MB = $\{b_{ij}\}$ profit robota 2. Profit robotów rozumiany jest jako suma preferencji osiągniętych przez nie podcelów:

$a_{ij} = l_{iR1} + l_{iR2}, b_{ij} = m_{iR1} + m_{iR2},$

gdzie:

l_{iR1} – profit robota 1, gdy osiągnął podcel przez wykonanie *i*-tej akcji,

l_{jR2} – profit robota 1, gdy osiągnął podcel przez wykonanie *j*-tej akcji przez robota 2,

m_{iR1} – profit robota 2, gdy osiągnął podcel przez wykonanie *i*-tej akcji przez robota 1,

m_{jR2} – profit robota 2, gdy osiągnął podcel przez wykonanie *j*-tej akcji.

Tabela 1

Macierz MA profity agenta 1

agent2 / agent1	stack D B	stack D F	stack E C	stack E D
stack C B	3	3+2	**(3)**	3+4
stack C F	0	2	0	4
stack A C	1	1+2	1	1+4
stack A D	0	2	0	4

Tabela 2

Macierz MB profity agenta 2

agent2 / agent1	stack D B	stack D F	stack E C	stack E D
stack C B	1	0	**(2)**	0
stack C F	1+3	3	2+3	3
stack A C	1	0	2	0
stack A D	1+4	4	2+4	4

Wiersze tabel przedstawiają możliwe realizacje operatorów STACK przez robota 1, wynikające ze znalezionego planu. Odpowiednio kolumny tabel przedstawiają możliwe akcje robota 2. Zatem, np. jeśli robot 1 zdecyduje się wykonać akcję {stack1 C} wynikającą z jego planu odpornego realizując podcel (on C B), to zyska profit l_{iR1} = 3 (pierwszy wiersz macierzy A). Profit ten może zostać zwiększony o 2, jeśli robot 2 wykona akcję realizującą podcel (on D F) – kolumna 2, albo zwiększony o 4, jeśli robot 2 wykona akcję realizującą podcel (on E D) – kolumna 4.

Przy założeniu hierarchiczności działań robotów proponuje się osiągnięcie kompromisu z wykorzystaniem równowagi von Stackelberga. Strategia ta doprowadzi do maksymalizacji sumy priorytetów wszystkich robotów w punktach równowagi.

Dla macierzy zysków przedstawionych powyżej, przy założeniu że robot 1 jest leaderem, a robot 2 followerem rozwiązaniem sytuacji konfliktowej jest wiersz 1 i kolumna 3. Wartości profitów robotów zaznaczono pogrubieniem w nawiasach.

Zastosowanie takiej strategii prowadzi do uprecyzyjnienia planu i sytuacji docelowej, w której każdy z agentów realizuje swoje cele w sensie równowagi von Stackelberga.

Plan precyzyjny rozwiązujący problem przybiera zatem postać:
step7 – ((((stack2 E)))
step6 – (((**pick-up2 E C**)) ((**stack1 A D**)))
step5 – (((**stack2 D F**)) ((unstack1 A)))
step4 – (((pick-up2 D)) ((**stack1 C B**)))
step3 – (((put-down2 D)) ((unstack1 C)))
step2 – (((pick-up2 D)) ((put-down1 B)))
step1 – (((unstack1 B)))),
natomiast stan docelowy, wynikający z równowagi von Stackelberga, to:
(and (on-table B) (on C B) (on E C) (clear C) (on-table F) (on D F) (on A D) (clear A))

3. Wnioski

Metoda zaproponowana w artykule pozwala na poszukiwania rozwiązania problemów planowania w środowisku wieloagentowym o strukturze hierarchicznej, w których nie występuje komunikacja pomiędzy agentami. Metoda ta wykorzystuje własności odwracalności niektórych systemów planowania w sztucznej inteligencji oraz metodologię poszukiwania rozwiązania w warunkach niejednoznacznej informacji o sytuacji początkowej problemu. Ponieważ znalezione rozwiązanie w ogólnym przypadku jest niekompletne (niedokładnie precyzuje akcje agentów), więc pokazano w jaki sposób sprecyzować plan wykorzystując elementy hierarchicznej teorii gier (równowagi niekooperacyjnej von Stackelberga).

Praca była finansowana z funduszu BK nr 227/RAu1/2015/1 Instytutu Automatyki Politechniki Śląskiej w 2015 roku.

Bibliografia

1. Kraus K., Sycara K.P., Evenchik A.: Reaching agreements through argumentation: a logical model and implementation. Artificial Intelligence, 104:1-69.
2. Gałuszka A., Świerniak A.: Planning in multi-agent environment as inverted STRIPS planning in the presence of uncertainty. Recent Advances In Computers, Computing and Communications (Ed. July 2002), WSEAS Press: 58-63.

3. Gałuszka A., Latawiec M.: Rozwiązywanie konfliktu pomiędzy robotami z wykorzystaniem elementów teorii gier. Zeszyty Naukowe Politechniki Śląskiej, s. Organizacja i Zarządzanie, z. 38, 2006, s. 35-41.
4. Nilson N.J.: Principles of Artificial Intelligence. Toga Publishing Company, Palo Alto, 1980, CA.
5. Bylander T.: The Computational Complexity of Propositional STRIPS Planning. Artificial Intelligence, 69, 1994, p. 165-204.
6. Mc Kinsey J.C.: Introduction to the Theory of Games. Mc Graw Hill, New York 1952.
7. Weld D.S., Anderson C.R., Smith D.E.: Extending Graphplan to Handle Uncertainty & Sensing Actions. Proc. 15th National Conf. on AI, 1998, 897-904.
8. Koehler J., Hoffmann J.: On Reasonable and Forced Goal Orderings and their Use in an Agenda-Driven Planning Algorithm. Journal of Artificial Intelligence Research, 12, 2000, p. 339-386.
9. Edelkamp S., Hoffmann J.: PDDL2.2. The language for the classical part of the 4th International Planning Competition. Tech. rep. 195, Albert-Ludwigs-Universitat Freiburg, Institut fur Informatik, 2004.
10. LaValle S.M.: Planning algorithms. Cambridge University Press, 2006.
11. Tao Peng, Shan Li.: Formation Control of Multiple Wheeled Mobile Robots via Leader-Follower Approach. Proc. 26th IEEE Chinese Control and Decision Conference (CCDC), 2014, p. 4215-4220.
12. Ćosić A., Šušić M, Graovac S, Katić D.: An Algorithm for Formation Control of Mobile Robots. Serbian Journal of Electrical Engineering, Vol. 10, No. 1, 2013, p. 59-72.
13. Madhevan B., Sreekumar M.: Tracking Algorithm Using Leader Follower Approach for Multi Robots. International Conference on Design and Manufacturing (IConDM2013), Procedia Engineering, Vol. 64, 2013, p. 1426-1435.

Abstract

The article presents multi-a. gent environment, in which each agent seeks to achieve its objective. It is assumed that the preferences of achieving the specific sub-goals which define the goal may be different for individual agents, and agent goals may be in conflict with each other. Additionally it is assumed a hierarchical structure among agents, due to different possibilities of their impact on the environment. The proposed method of finding the solution combines the properties of invertability of planning problems in artificial intelligence and a methodology to seek a solution under ambiguous information about the initial situation of the problem. Because the solution found in the general case is incomplete (agents' actions are imprecise), a method of finding the plan using elements of the theory of hierarchical games (non-cooperative Stackelberg equilibrium) is proposed.