ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2006
Seria: AUTOMATYKA z. 146 Nr kol. 1739

Adam GAŁUSZKA
Politechnika Śląska

ANALIZA ŚWIATA KLOCKÓW JAKO ZADANIE PROGRAMOWANIA LINIOWEGO

Streszczenie. W pracy sprowadzono problem planowania w środowisku tzw. świata klocków, posiadający reprezentację STRIPS, do problemu programowania liniowego. Uwzględniono również niepełność dostępnej informacji i przeanalizowano jej wpływ na rozwiązanie zadania programowania liniowego. Pokazano także zależność rozmiaru problemu liniowego od rozmiaru świata klocków. Transformacja do zadania programowania liniowego polega na założeniu liczby etapów planowania i przypisaniu każdemu warunkowi i operatorowi systemu STRIPS w każdym etapie planowania jednej zmiennej. Przy ograniczeniu dopuszczalnych wartości zmiennych do przedziału <0,1> założono, że wartości te odpowiadają stopniu spełnienia warunków oraz stopniu zastosowania operatorów w systemie STRIPS.

BLOCK WORLD ANALYSIS AS LINEAR PROGRAMMING PROBLEM

Summary. In the paper block world environment with STRIPS representation is presented as linear programming problem. It is also assumed that information about state of the problem may be incomplete and its influance to linear programming problem solution is shown. Translation of the STRIPS planning to linear programming is based on assumption of the number of planning steps and mapping conditions and oparators in each planning step to variables. Variables are mapped to values between 0 and 1: it corresponds to truth degree of conditions and operators.

1. Wprowadzenie

Próby implementacji robota inteligentnego pociągają za sobą konieczność analizy i rozwiązywania takich zadań jak percepcja i rozumienie otoczenia, osiąganie celu, planowanie

zadań, wykonywanie zadań i monitorowanie ruchów robota. Przez robot inteligentny rozumiemy raczej robot autonomiczny, zdolny do analizy otoczenia, planowania i wykonywania ruchów w otoczeniu niezdeterminowanym (Yen i inni 1995). Jądrem implementacji takiego robota jest system, w którym zapisana wiedza oraz określona metoda poszukiwania rozwiązań problemów pozwala na generowanie planu ruchów robota.

W ogólnym przypadku problem ten jest bardzo złożony. Wynika to ze złożoności i sposobu monitorowania otoczenia robota, liczby stopni swobody robota wynikających z jego konstrukcji mechanicznej, liczby poziomów sterowania robotem, złożoności inteligentnego oprogramowania. Wszystko to spowodowało, że inteligentna robotyka stała się dziedziną interdyscyplinarną (Yen i inni 1995, Popovic i Bhatkar 1994). Praktyczne implementacje inteligentnych systemów wskazują, że zarówno procesy planowania zadań, jak i planowania trajektorii powinny uwzględniać niepewność dostępnej informacji (Yen i inni 1995, Ruspini i inni 1995).

Obecnie problemy poprawy efektywności planowania z wykorzystaniem reprezentacji STRIPS oraz planowania w warunkach niepewności są często poruszane w literaturze (np.: Slaney i Thiebaux 2001, Howe i Dahlman 2002, Rintanen 1999, Weld 1999, Weld i inni 1998, Sierocki 1997, Joslin i Roach 1989/1990, Nebel i Koehler 1995, Świerniak i Gałuszka 1999).

W niniejszej pracy rozważane są problemy planowania zadań i osiągania celów w środowisku tzw. świata klocków. W szczególności analizowane są: sposób redukcji złożoności obliczeniowej oraz modelowanie planowania jako systemu zawierającego niepewną (niejednoznaczną) informację o stanie początkowym problemu. Tego typu problemy mogą być reprezentowane za pomocą systemu STRIPS (Gałuszka 1999).

W pracy sprowadzono problem planowania STRIPS w środowisku tzw. świata klocków do problemu programowania liniowego według heurystyki Bylandera (1997). Takie podejście pozwala efektywniej rozwiązać problemy rzeczywistych (dużych) rozmiarów, jednak przy założeniu, że otrzymane rozwiązanie posiada interpretację w świecie klocków. Spełnienie tego założenia redukuje klasę złożoności obliczeniowej problemu planowania. Dzieje się tak dlatego, że:

- problem planowania optymalnego (znalezienia planu o najmniejszej liczbie operatorów) w świecie klocków jest NP zupełny (Gupta i Nau 1992);

- transformacja do zadania programowania liniowego jest wielomianowa (tzn. czas transformacji jest ograniczony wielomianem zależnym od rozmiaru problemu);
- zadanie programowania liniowego jest problemem wielomianowo złożonym (należy do klasy P złożoności obliczeniowej).

Ponieważ rozwiązania otrzymane z zadania programowania liniowego (ZPL) są wartościami z przedziału <0,1>, a wartości te odpowiadają stopniu spełnienia warunków oraz stopniu zastosowania operatorów w systemie STRIPS („0" odpowiada niespełnienie warunku i niezastosowanie operatora, „1" odpowiada całkowite spełnienie warunku i zastosowanie operatora), zaproponowana metodologia może służyć analizie problemów STRIPS z uwzględnieniem niepewności. Niepewność w problemach planowania typu STRIPS jest często reprezentowana jako niejednoznaczność sytuacji początkowej problemu, tzn. zakładamy, że mamy do czynienia z alternatywą możliwych sytuacji początkowych (Smith i Weld 1998, Weld i inni 1998). Takie założenie uczyniono w pracy.

Oryginalnym wkładem autora jest sprowadzenie problemu planowania STRIPS w środowisku tzw. ograniczonego świata klocków do problemu programowania liniowego i analiza wpływu niepełności informacji na rozwiązanie zadania liniowego, jak również pokazanie zależności rozmiaru problemu liniowego od rozmiaru świata klocków. Dla uproszczenia analizy transformacji do ZPL zamieszczone w pracy wyniki symulacji odnoszą się jedynie do problemu dekompozycji świata klocków, dlatego też analizowane środowisko zostało nazwane środowiskiem ograniczonym.

2. Problem planowania typu STRIPS

Problem planowania STRIPS (*STRIPS planning problem*) składa się z czterech zbiorów $\{P, O, I, G\}$ (Nilson 1980, Bylander 1984), gdzie :

- P jest skończonym zbiorem podstawowych formuł, zwanych warunkami (*conditions*),
- O jest skończonym zbiorem operatorów, gdzie każdy operator $o \in O$ przyjmuje formę $o^+, o^- \Rightarrow o_+, o_-$, gdzie
  - $o^+ \subseteq P$ są pozytywnymi warunkami stosowalności operatora (*positive preconditions*),
  - $o^- \subseteq P$ są negatywnymi warunkami stosowalności operatora (*negative preconditions*),

- $o_+ \subseteq P$ są pozytywnymi skutkami zastosowania operatora (*positive postconditions*),
- $o_- \subseteq P$ są negatywnymi skutkami zastosowania operatora (*negative postconditions*),
- $I \subseteq P$ jest stanem początkowym zadania (*initial state*),
- $G = \{G_+, G_-\}$ jest spełnialną koniunkcją pozytywnych i negatywnych warunków, w której $G_+ \subseteq P$ są pozytywnymi warunkami, a $G_- \subseteq P$ są negatywnymi warunkami.

Każdy stan problemu planowania może być opisany jako podzbiór $S \subseteq P$ w taki sposób, że warunek $p \in P$ jest spełniony (prawdziwy) w opisie stanu, jeśli $p \in S$ i niespełniony (fałszywy) w przeciwnym przypadku. Zbiór I precyzuje, które warunki są prawdziwe, a które fałszywe w stanie początkowym problemu, tzn. warunek $p \in P$ jest prawdziwy w stanie początkowym, jeśli $p \in I$ lub fałszywy w przeciwnym przypadku. Zbiór G precyzuje cel, tzn. $S \subseteq P$ jest stanem docelowym problemu, jeśli S spełnia G, tzn. jeśli $G_+ \subseteq S$ i $G_- \cap S = \emptyset$.

Efekt zastosowania skończonej sekwencji operatorów do stanu S może być opisany poprzez funkcję *Result*, która jest zdefiniowana następująco (symbol "\" oznacza różnice zbiorów):

$$Result(S, \{\}) = S,$$

$$Result(S, \{o\}) = \begin{cases} (S \cup o_+) \setminus o_-, & \text{jeśli } o^+ \subseteq S \wedge o^- \cap S = \emptyset; \\ S, & \text{w przeciwnym przypadku,} \end{cases}$$

$$Result(S, \{o_1, o_2, \ldots, o_n\}) = Result(Result(S, \{o_1\}), \{o_2, \ldots, o_n\}).$$

Z definicji funkcji *Result* wynika, że każdy operator może być zastosowany do każdego stanu, natomiast tylko ten operator daje efekt, dla którego spełnione są warunki stosowalności (*positive and negative preconditions*). Dany operator może występować wiele razy w sekwencji operatorów tworzących plan.

Skończona sekwencja operatorów $\{o_1, o_2, \ldots, o_n\}$ jest rozwiązaniem problemu planowania, jeśli stan wynikający z zastosowania funkcji $Result(I, \{o_1, o_2, \ldots, o_n\})$ jest stanem docelowym problemu.

Jako współczesne zastosowania problemów typu „świat klocków" posiadających reprezentację STRIPS można wymienić:

- środowisko testowe (benchmark) dla problemów planowania (odpowiednik problemu komiwojażera dla algorytmów przeszukiwania grafu) (Howe i Dahlman 2002);
- reprezentacja dla problemów logistycznych (magazynowania, planowania dostaw) - przesuwanie klocków to przesuwanie pakunków, ciężarówek, samolotów itp.) (Slaney J. i S. Thiebaux 2001);
- reprezentacja dla problemów załadunku (dla świata klocków z ograniczonym miejscem na stole) (Slavin 1996).

3. Transformacja do zadania programowania liniowego

Idea transformacji do ZPL polega na przypisaniu każdemu warunkowi i operatorowi systemu STRIPS w każdym etapie planowania zmiennej (Bylander 1997). Zakładamy, że jeśli *p* jest warunkiem i jeśli planowanie jest podzielone na *l* etapów, to potrzebujemy *l*+1 zmiennych dla tego warunku: *p(0), p(1), ..., p(l).* Jeśli z kolei *op* będzie operatorem, to potrzebujemy *l* zmiennych dla każdego operatora: *op(0), op(1), ..., op(l-1).* Funkcja celu osiąga wartość maksymalną, jeśli wartości warunków definiujących cel w ostatnim etapie planowania są równe 1.

Rozważmy przykład ze środowiska świata klocków (Gałuszka i Świerniak 2004). Ograniczmy problem jedynie do dekompozycji stanu początkowego pewnej konfiguracji klocków. Dla prostoty założono następującą prostą konfigurację klocków:

on(A,B) ^ on(D,C)

Dysponujemy jednym operatorem umożliwiającym dekompozycję:

move-to-table(x,y) – oznacza ściągnięcie klocka *x* z *y* i położenie na stole
preconditions: on(x,y), clear(x)
postconditions: not(on(x,y)), clear(y)

Stan będzie zdekomponowany, jeśli dla każdego klocka będzie prawdziwy warunek:

clear(A), clear(B), clear(C), clear(D)

Załóżmy 2 etapy planowania (dla każdej z możliwych założonych sytuacji

$P = \{on(x,y), clear(x)\}$

$O = \{move-to-table(x,y) : on(x,y) \wedge clear(x) \Rightarrow not(on(x,y)) \wedge clear(y)\}$

$I = \{on(A,B), on(D,C), clear(A), clear(D)\}$

$G = \{clear(A) \wedge clear(B) \wedge clear(C) \wedge clear(D)\}$

początkowych wystarczą 2 operatory do ich dekompozycji). Wtedy system STRIPS dla problemu dekompozycji jest zdefiniowany poniżej:

Tak więc mamy (16+16+16) zmiennych dla warunków:

1. *on(A,B)(i)*
2. *on(A,C)(i)*
3. *on(A,D)(i)*
4. *on(B,A)(i)*
5. *on(B,C)(i)*
6. *on(B,D)(i)*
7. *on(C,A)(i)*
8. *on(C,B)(i)*
9. *on(C,D)(i)*
10. *on(D,A)(i)*
11. *on(D,B)(i)*
12. *on(D,C)(i)*
13. *clear(A)(i)*
14. *clear(B)(i)*
15. *clear(C)(i)*
16. *clear(D)(i)* $i = 0, 1, 2$

Dodatkowo potrzebujemy 24 zmienne dla operatorów (12+12):

1. *move-to-table(A,B)(i)*
2. *move-to-table(A,C)(i)*
3. *move-to-table(A,D)(i)*
4. *move-to-table(B,A)(i)*
5. *move-to-table(B,C)(i)*
6. *move-to-table(B,D)(i)*
7. *move-to-table(C,A)(i)*
8. *move-to-table(C,B)(i)*
9. *move-to-table(C,D)(i)*
10. *move-to-table(D,A)(i)*
11. *move-to-table(D,B)(i)*
12. *move-to-table(D,C)(i)*

Funkcja celu definiuje stan docelowy:

$$Max\ (\ clear(A)(2) + clear(B)(2) + clear(C)(2) + clear(D)(2)\ )$$

Dla rozwiązanego problemu planowania funkcja celu będzie równa 4 (4 warunki będą prawdziwe).

Co najwyżej 1 operator może być zastosowany na każdym etapie:

$$\Sigma move\text{-}to\text{-}table(x,y)(1) \leq 1$$
$$\Sigma move\text{-}to\text{-}table(x,y)(2) \leq 1$$

Operator nie może być zastosowany, jeśli jego warunki stosowalności nie są prawdziwe:

$on(x,y)(i) \geq move\text{-}to\text{-}table(x,y)(i)$ - dla wszystkich 12 operatorów w i-tym etapie;
$clear(A)(i) \geq move\text{-}to\text{-}table(A,B)(i) + move\text{-}to\text{-}table(A,C)(i) + move\text{-}to\text{-}table(A,D)(i)$
$clear(B)(i) \geq move\text{-}to\text{-}table(B,A)(i) + move\text{-}to\text{-}table(B,C)(i) + move\text{-}to\text{-}table(B,D)(i)$
$clear(C)(i) \geq move\text{-}to\text{-}table(C,A)(i) + move\text{-}to\text{-}table(C,B)(i) + move\text{-}to\text{-}table(C,D)(i)$
$clear(D)(i) \geq move\text{-}to\text{-}table(D,A)(i) + move\text{-}to\text{-}table(D,B)(i) + move\text{-}to\text{-}table(D,C)(i)$
$i = 0, 1.$

Następna grupa ograniczeń opisuje zmiany wartości zmiennych dla warunków w następnych etapach po zastosowaniu operatorów. Są to ograniczenia równościowe:

$$clear(A)(i+1) =$$
$$= clear(A)(i) + move\text{-}to\text{-}table(B,A)(i) + move\text{-}to\text{-}table(C,A)(i) + move\text{-}to\text{-}table(D,A)(i)$$

i tak dla wszystkich pozostałych warunków: clear(x)(i+1). Potrzebne są jeszcze równania opisujące zmiany warunków typu: on(x,y):

$$on(A,B)(i+1) = on(A,B)(i) - move\text{-}to\text{-}table(A,B)(i)$$

Wreszcie ograniczenia na wartości zmiennych: wartości te muszą zawierać się w przedziale <0,1> odpowiadającym wartościom prawdy dla danych warunków.

4. Wyniki symulacji

Otrzymany w wyniku przedstawionej transformacji problem ZPL ma rozmiar (dla „*n* " klocków i „*l* " kroków planowania, gdzie zawsze $l \leq n$ dla problemu dekompozycji):

- liczba zmiennych: mniej niż $2l(n^2-1)$;
- liczba ograniczeń nierównościowych: $l\,(n^2 + n + 1)$;
- liczba ograniczeń równościowych: $l\,(n^2 + n)$.

Dla ilustracji zagadnienia transformacji zaimplementowano w środowisku MATLAB 2 przykłady. W pierwszym przypadku był to problem z kompletną informacją o transformowanym środowisku, w przypadku drugim – z informacją niekompletną. W obydwu przypadkach po transformacji do ZPL otrzymujemy problem o 72 zmiennych, 34 ograniczeniach nierównościowych i 32 równościowych.

Pierwszy z nich to dekompozycja świata klocków o sytuacji początkowej danej jako:

on(A,B), on(D,C), clear(A), clear(D)

Rozwiązaniem ZPL jest zbiór wartości zmiennych, które dają optymalną wartość funkcji celu (tabl.1). Wektor 72 zmiennych ZPL podzielony został w tabeli na 5 kolumn, które kolejno odpowiadają za warunki (kolumny 1, 3 i 5) oraz operatory (kolumny 2 i 4) zgodnie z transformacją z punktu 3. Wartości te korespondują z wartościami prawdy warunków i operatorów:

move-to-table(D,C) (0)=1 i *move-to-table(A,B) (1)=1*,

które rozwiązują problem dekompozycji w 2 krokach.

Przypadek drugi reprezentuje problem z niejednoznaczną informacją o stanie początkowym Teraz sytuacja początkowa składa się z dwóch możliwych stanów początkowych:

(on(A,B), on(D,C), clear(A), clear(D)) or (on(A,C), on(D,B), clear(A), clear(D))

W ZPL wartości zmiennych odpowiadające predykatom:

on(A,B)(0), on(D,C)(0), on(A,C)(0), on(D,B)(0)

są równe 0.5, co reprezentuje przypadek niejednoznaczności następującej sytuacji: czy blok *A* jest na *C* lub *B*, a blok *D* jest na *C* lub *B*.

Tablica 1

Rozwiązanie ZPL w przypadku 1

Warunki i = 0	Operatory i = 0	Warunki i = 1	Operatory i = 1	Warunki i = 2
1: 1.0000	1: -0.0000	1: 1.0000	1: 1.0000	1: 0.0000
2: 0.0000	2: -0.0000	2: 0.0000	2: 0.0000	2: -0.0000
3: 0.0000	3: 0.0000	3: 0.0000	3: 0.0000	3: 0.0000
4: 0.0000	4: 0.0000	4: 0.0000	4: 0.0000	4: -0.0000
5: -0.0000	5: -0.0000	5: -0.0000	5: -0.0000	5: -0.0000
6: -0.0000	6: 0.0000	6: 0.0000	6: 0.0000	6: 0.0000
7: 0.0000	7: 0.0000	7: -0.0000	7: -0.0000	7: 0.0000
8: -0.0000	8: 0.0000	8: 0.0000	8: -0.0000	8: -0.0000
9: 0.0000	9: -0.0000	9: 0.0000	9: 0.0000	9: 0.0000
10: 0.0000	10: 0.0000	10: -0.0000	10: -0.0000	10: 0.0000
11: -0.0000	11: 0.0000	11: -0.0000	11: -0.0000	11: 0.0000
12: 1.0000	12: 1.0000	12: 0.0000	12: -0.0000	12: 0.0000
13: 1.0000		13: 1.0000		13: 1.0000
14: -0.0000		14: 0.0000		14: 1.0000
15: 0.0000		15: 1.0000		15: 1.0000
16: 1.0000		16: 1.0000		16: 1.0000

Rozwiązanie ZPL w przypadku drugim daje wartości prawdy dla operatorów:

move-to-table(A,B)(0)=0.5
move-to-table(A,C)(0)=0.5
move-to-table(D,B)(1)=0.5
move-to-table(D,C)(1)=0.5,

które dają rozwiązanie problemu również w dwóch krokach (tabl.2). Interpretacja wartości prawdy jest następująca: w kroku pierwszym ściągnąć blok A na stół (z bloku B lub C), następnie ściągnąć blok D z B lub C.

Powyższe przykłady pokazują przydatność transformacji problemów planowania w przypadku problemów z pewną i niepewną informacją. Nie wiadomo jednak, jak duża jest grupa problemów planowania, dla których rozwiązanie problemu po transformacji daje rezultaty, które można poprawnie zinterpretować w danej dziedzinie (tutaj – świat klocków). Określenie tej grupy będzie tematem dalszych prac.

Tablica 2

Rozwiązanie ZPL w przypadku 2

Warunki i = 0	Operatory i = 0	Warunki i = 1	Operatory i = 1	Warunki i = 2
1: 0.5000	1: 0.5000	1: 0.0000	1: 0.0000	1: 0.0000
2: 0.5000	2: 0.5000	2: -0.0000	2: -0.0000	2: -0.0000
3: -0.0000	3: -0.0000	3: 0.0000	3: 0.0000	3: 0.0000
4: 0.0000	4: 0.0000	4: 0.0000	4: 0.0000	4: -0.0000
5: -0.0000	5: -0.0000	5: -0.0000	5: -0.0000	5: -0.0000
6: 0.0000	6: -0.0000	6: 0.0000	6: -0.0000	6: 0.0000
7: -0.0000	7: -0.0000	7: 0.0000	7: 0.0000	7: -0.0000
8: -0.0000	8: -0.0000	8: 0.0000	8: 0.0000	8: -0.0000
9: -0.0000	9: 0.0000	9: 0.0000	9: -0.0000	9: 0.0000
10: -0.0000	10: -0.0000	10: -0.0000	10: 0.0000	10: -0.0000
11: 0.5000	11: 0.0000	11: 0.5000	11: 0.5000	11: 0.0000
12: 0.5000	12: 0.0000	12: 0.5000	12: 0.5000	12: 0.0000
13: 1.0000		13: 1.0000		13: 1.0000
14: -0.0000		14: 0.5000		14: 1.0000
15: -0.0000		15: 0.5000		15: 1.0000
16: 1.0000		16: 1.0000		16: 1.0000

5. Podsumowanie i wnioski

Redukcja do ZPL pozwala efektywniej rozwiązać problemy rzeczywistych (dużych) rozmiarów, jednak przy założeniu, że otrzymane rozwiązanie posiada interpretację w świecie klocków. Spełnienie tego założenia redukuje klasę złożoności obliczeniowej problemu planowania. Dzieje się tak dlatego, że:

- problem planowania optymalnego w świecie klocków jest NP zupełny;
- transformacja do zadania programowania liniowego jest wielomianowa (dla n klocków potrzebujemy mniej niż $l(n^2+n+n^2)$ kroków);
- zadanie programowania liniowego jest problemem wielomianowo złożonym.

Programowanie całkowitoliczbowe da w wyniku wartości zmiennych 0 lub 1, co odpowiadałoby rozwiązaniu problemu planowania bez uwzględniania niepewności co do warunków w stanie początkowym problemu. Podejście takie jest jednak nieefektywne, gdyż problem programowania całkowitoliczbowego jest NP-zupełny, w wyniku czego redukowalibyśmy problem NP-zupełny do innego NP-zupełnego, co nie jest postępem z

punktu widzenia zwiększania efektywności obliczeniowej. Programowanie liniowe da w wyniku wartości z przedziału <0,1> włącznie, co pozwala uwzględniać niepewność co do stopnia prawdy warunków w stanie początkowym problemu, jednak nie zawsze otrzymany wynik daje poprawną interpretację znalezionego planu. Jest to koszt, jaki ponosimy redukując problem planowania do wielomianowego.

W pracy zamodelowano ZPL odpowiadające środowisku świata klocków ograniczonego do problemów dekompozycji w programie MATLAB. Pokazano, że redukcja do ZPL w tym przypadku jest wielomianowa.

Niniejsza praca była finansowana z funduszu BK-208/RAu1/2006 temat 1.

LITERATURA

1. Barret A. i D.S. Weld. 1994. Partial-order planning: evaluating possible efficiency gains. Artificial Intelligence, 67:71-112.
2. Bylander, T. 1994. The computational complexity of propositional STRIPS planning. Artificial Intelligence, 69:165-204.
3. Bylander T. 1997. A Linear Programming Heuristic for Optimal Planning. Int. Conf. American Association for Artificial Intelligence, 1997, www.aaai.org.
4. Cormen T.H., Ch.E. Leiserson, R.L. Rivest. 1994. Introduction to Algorithms. The Massachusetts Institute of Technology.
5. Dougherty, E.R. i Ch.R. Giardina. 1988. Mathematical Methods for Artificial Intelligence and Autonomous Systems. Prentice-Hall International, Inc. USA.
6. Gałuszka A. 1999. „Distinguish- and indistinguishability of STRIPS planning problem elements". In Proceedings of the 11th European Simulation Symposium (Erlangen, 1999), 95-99.
7. Gałuszka A. i A. Świerniak. 2004. Translation STRIPS Planning in Multi-Robot Environment to Linear Programming. LNCS, Springer-Verlag, Volume 3070 / 2004, pp.768-773.
8. Garey M.R. i D.S. Johnson. 1979. Computers and Intractability - A Guide to the Theory of NP-Completeness. Freeman, San Francisco, CA.

9. Gupta N. i Nau D.S. 1992. On the complexity of Blocks-World Planning. Artificial Intelligence, 56 (2-3): 223-254.
10. Howe A.E. i E.Dahlman. 2002. A Critical Assesment of Benchmark Comparison in Planning. Journal of Artificial Intelligence Research, 17: 1-33.
11. Joslin D. i J. Roach. 1989/90. A Theoretical Analysis of Conjuctive-Goal Problems. Artificial Intelligence, 41: 97-106.
12. Klir, G.J., T.A. Folger. 1988. Fuzzy Sets, Uncertainty, and Information. Prentice-Hall International, Inc. USA.
13. McDermott D. i J. Hendler. 1995. Planning: what it is, what it could be, an introduction to the special issue on planning and scheduling. Artificial Intelligence, 76:1-16.
14. Nebel B. i J. Koehler. 1995. Plan reuse versus plan generation: a theoretical and empirical results. Artificial Intelligence, 76: 427-454.
15. Nilson N. J. 1980. Principles of Artificial Intelligence. Toga Publishing Company, Palo Alto, California.
16. Popovic, D. i V.P. Bhatkar. 1994. Methods And Tools For Applied Artificial Intelligence. Marcel Dekker, Inc., New York, NY.
17. Rintanen, J. 1999. Constructing Conditional Plans by a Theorem-Prover. Journal of Artificial Intelligence Research, 10: 323-352.
18. Ruspini, E.H., A. Saffiotti, K. Konolige. 1995. Progress in Research on Autonomous Vehicle Motion Planning. [in] Yen, J.; R. Langari; L.A. Zadeh. 1995. Industrial Applications of Fuzzy Logic and Intelligent Systems. IEEE Press. New York.
19. Sierocki I. 1997. A Serial Decomposition of Planning Problems. Proc. Fourth International Symposium on Methods and Models in Automation and Robotics. Międzyzdroje, Poland, 1179-1184.
20. Slaney J. i S. Thiebaux. 2001. Block World revisited. Artificial Intelligence, 125: 119-153.
21. Slavin T. 1996. Virtual port of call, in: New Scientist, June, 40-43.
22. Smith, D.E. i D.S. Weld. 1998. Conformant Graphplan. Proc. 15th National Conf. on AI.

23. Świerniak, A. i A. Gałuszka. 1999. Betweenness and Indistinguishability in Modeling and Control of Uncertain Systems, Proc. of 18th IASTED Conference Modeling, Identification and Control, Innsbruck, 56-58.

24. Weld, D.S. 1999. Recent Advantages in AI Planning. AI Magazine.

25. Weld, D.S., C.r. Anderson i D.E. Smith. 1998. Extending Graphplan to Handle Uncertainty & Sensing Actions. Proc. 15th National Conf. on AI.

26. Yen, J., R. Langari, L.A. Zadeh. 1995. Industrial Applications of Fuzzy Logic and Intelligent Systems. IEEE Press. New York.

Recenzent: Prof. dr hab. inż. Zbigniew Banaszak

Abstract

In the paper block world environment with STRIPS representation is considered. This domain is often used to model planning problems because of complex action definition and simple physical interpretation. Block world today stated an experimentation benchmark for planning algorithms. Also more realistic situations can be presented as Block World problems, where moving blocks corresponds to moving different objects like packages, trucks and planes. The case of Block World problem where the table has a limited capacity corresponds to a container-loading problem.

Planning with complete information is usually at least NP-complete problem (e.g. optimal planning in Block World environment is NP-complete). Planning in the presence of incompleteness belongs to the next level in the hierarchy of completeness. It is assumed that incompleteness is when in the problem there are some possible initial situations and one goal state. To reduce computational complexity of this approach a transformation to Linear Programming problem is proposed. The transformation from planning to Linear Programming is based on mapping of conditions and operators in each plan step to variables. Truth-values of conditions are mapped to 0 and 1 for the planning without incompleteness, and to any values between 0 and 1 for planning with incomplete information. The objective function reaches the maximum if the goal situation is true in last step of planning. Simulations illustrate the reduced problem.