ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1998
Seria: AUTOMATYKA z. 125 Nr kol. 1391

Adam GAŁUSZKA
Politechnika Śląska

ZACHŁANNY ALGORYTM PLANOWANIA W „ŚWIECIE KLOCKÓW"

Streszczenie. Niniejszy artykuł analizuje zadania ze „świata klocków", w których sytuację docelową stanowi opis wielu wież klocków. W ogólnym przypadku generacja optymalnego planu rozwiązującego takie zadanie jest problemem NP złożonym. Graf ograniczeń kolejnościowych redukuje przestrzeń stanów zadania. Artykuł pokazuje, jak taki graf można zbudować. Graf ten umożliwia zaimplementowanie algorytmu zachłannego, który generuje rozwiązanie. Algorytm ten jest wielomianowo złożony w czasie. Omówione zostały także warunki konieczne i wystarczające optymalności algorytmu zachłannego.

GREEDY ALGORITHM IN BLOCK WORLD

Summary. In this paper block world instance where the goal state is a complete description of a set of stacks is presented. In general to generate an optimal plan for such problem is NP- hard. Precedence constraints graph reduces block world states space. It is shown how this graph can be built. Now it is possible to implement greedy algorithm which generates solution. This algorithm is polynomial-time complete. Necessary and sufficient conditions of greedy algorithm optimality are also discussed.

1. Wstęp

Problemy planowania należą do najważniejszych i najczęściej rozpatrywanych problemów sztucznej inteligencji. Klasyczny problem planowania zadań robota polega na przekształceniu bieżącego stanu świata do pożądanego, docelowego stanu świata. Jedną z pierwszych reprezentacji problemów planowania (rok 1960) był tzw. system STRIPS [4], który do dnia dzisiejszego wydaje się być najbardziej popularny [7]. Istnieje wiele różnych metod służących do rozwiązywania problemów planowania [3, 5, 7, 8, 9]. Problem złożoności obliczeniowej tych algorytmów jest także często poddawany dyskusji [3, 8, 9]. Zwykle algorytmy rozwiązujące problemy planowania są NP złożone.

Artykuł ten prezentuje zastosowanie algorytmu zachłannego [2] do rozwiązania klasycznego problemu planowania zadań robota. Algorytmy zachłanne często znajdują zastosowanie w teorii grafów do optymalnego przeszukiwania ścieżek w grafie [2], np. do rozwiązania problemu minimalnego drzewa rozpinającego. Prezentowany algorytm jest

wielomianowo złożony. Artykuł wyjaśnia także, dlaczego strategia zachłanna redukuje złożoność obliczeniową zadania.

2. Reprezentacja „świata klocków" w systemie STRIPS

W przypadku ogólnym, zadanie ze „świata klocków" jest reprezentowane przez trzy listy $<\vartheta,\Sigma,\Omega>$ [4]:

- zbiór operatorów (ϑ);
- zbiór predykatów opisujących początkowy stan świata w zadaniu (Σ);
- zbiór predykatów opisujących docelowy stan świata w zadaniu (Ω).

Predykaty opisujące stan początkowy i docelowy powinny opisywać jedną, fizycznie możliwą konfigurację klocków względem siebie. Ponadto opis stanu początkowego powinien być kompletny, tzn. powinien zawierać każdy prawdziwy predykat wynikający z początkowego ułożenia klocków.

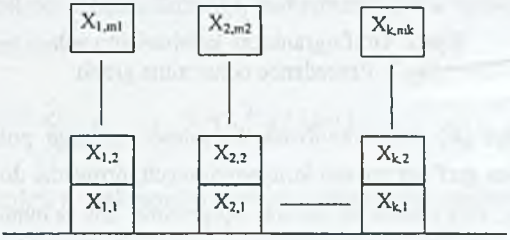
Rezultatem pracy algorytmu jest zbiór operatorów przekształcających stan początkowy w stan docelowy świata z zadania. Optymalnym rezultatem jest osiągnięcie stanu docelowego w minimalnej liczbie kroków, tzn. przy użyciu minimalnej liczby operatorów. Operatory w systemie STRIPS składają się z trzech podlist: listy warunków stosowalności *(precondition list)*, listy skreśleń *(delete list)* oraz listy dopisków *(add list)*. Lista warunków stosowalności jest zbiorem predykatów, które muszą być prawdziwe w opisie bieżącego stanu świata, aby móc zastosować dany operator. Lista skreśleń jest zbiorem predykatów, które przestają być prawdziwe po zastosowaniu danego operatora, natomiast lista dopisków to zbiór predykatów, które stają się prawdziwe po zastosowaniu danego operatora. Innymi słowy można powiedzieć, że ostatnie dwie listy opisują efekt zastosowania operatora do bieżącego stanu świata w zadaniu.

Załóżmy, że istnieją cztery operatory, które pozwalają przekształcać stan świata w zadaniu:

1) **pickup(x)** - oznacza podniesienie klocka X ze stołu;
   Lista warunków stosowalności: ONTABLE(X), CLEAR(X), HANDEMPTY
   Lista skreśleń: ONTABLE(X), CLEAR(X), HANDEMPTY
   Lista dopisków: HOLDING(X)
2) **putdown(x)** - oznacza położenie klocka X na stół;
   Lista warunków stosowalności: HOLDING(X)
   Lista skreśleń: HOLDING(X)
   Lista dopisków: ONTABLE(X), CLEAR(X), HANDEMPTY
3) **stack(x,y)** - oznacza położenie klocka X na klocku Y;
   Lista warunków stosowalności: HOLDING(X), CLEAR(Y)
   Lista skreśleń: HOLDING(X), CLEAR(Y)
   Lista dopisków: HANDEMPTY, ON(X,Y), CLEAR(X)
4) **unstack(x,y)** - oznacza ściągnięcie klocka X z klocka Y;

Lista warunków stosowalności: HANDEMPTY, CLEAR(X), ON(X,Y)
Lista skreśleń: HANDEMPTY, CLEAR(X), ON(X,Y)
Lista dopisków: HOLDING(X), CLEAR(Y)

Przypadek ogólny zadania ze „świata klocków", w którym celem jest ułożenie zbioru wież, przedstawia rysunek 1. Stan początkowy zadania może być dowolną kombinacją wszystkich klocków występujących w opisie sytuacji docelowej.



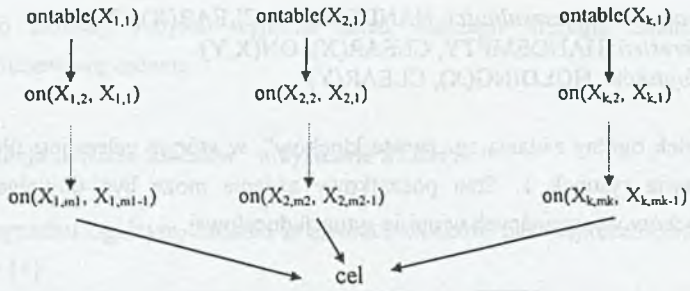
Rys.1. Zadanie ze „świata klocków" - stan docelowy
Fig.1. General block world instance - goal state

- k jest liczbą wież;
- m_i dla i = 1, 2 ... k jest liczbą klocków w i-tej wieży;
- $\sum_{i=1}^{k} m_k = n$ jest liczbą wszystkich klocków w zadaniu.

Łatwo można sobie wyobrazić, że graf przestrzeni stanów zadania, który opisuje wszystkie możliwe stany zadania pomiędzy stanami początkowym i docelowym, jest bardzo skomplikowany. Praca [3] pokazuje, że problem generacji optymalnego planu przez przeszukiwanie grafu przestrzeni stanów zadania może być zredukowany do NP złożonego problemu 3-wymiarowego dopasowania [1] poprzez transformację wielomianową.

3. Graf ograniczeń kolejnościowych

Bezpośrednio z charakterystycznej formy sytuacji docelowej zadania z rysunku 1 wynika graf ograniczeń kolejnościowych przedstawiony na rysunku 2. Pokazuje on, które z podceli muszą być wykonane przed innymi, aby osiągnąć pożądany stan docelowy. Graf ten ponadto musi zawierać predykaty ontable(X). Brak rozszerzenia opisu sytuacji docelowej o predykaty ontable(X) powoduje, że stan docelowy zadania staje się nieszeregowalny [5]. Może to uniemożliwić uszeregowanie podceli w sposób prowadzący do rozwiązania zadania. Sytuacja taka znana jest jako anomalia Sussmana. [5, 6, 10].



Rys.2. Graf ograniczeń kolejnościowych
Fig.2. Precedence constraints graph

Należy zauważyć [6], że jakakolwiek kolejność realizacji podcelów, która spełnia warunki narzucone przez graf ograniczeń kolejnościowych, prowadzi do generacji operatorów rozwiązujących zadanie. Aby rozwiązać zadanie optymalnie, tzn. w minimalnej liczbie kroków, należy jednak wybrać określoną kolejność realizacji podcelów. Kolejność ta może być generowana za pomocą strategii zachłannej.

4. Algorytm zachłanny

Algorytm zachłanny podejmuje decyzje o wyborze kolejnego podcelu do realizacji według następującego schematu:

- istnieje k różnych podcelów, które można wybrać do realizacji w każdym kroku (k jest liczbą wież); wynika to bezpośrednio z grafu ograniczeń kolejnościowych;
- spośród nich wybierz ten podcel, do którego osiągnięcia potrzeba najmniejszej liczby operatorów;
- jeśli liczba operatorów potrzebnych do osiągnięcia pojedynczego podcelu jest taka sama dla kilku podcelów, wybierz pierwszy z nich.

Liczba operatorów potrzebna do osiągnięcia pojedynczego podcelu (L) może być łatwo obliczona z następującej formuły:

- jeśli wybrany podcel ma postać ontable(X), wtedy:

$$L = (\Sigma \text{ wszystkich klocków na klocku X w opisie stanu bieżącego} * 2) + 2$$

- jeśli wybrany podcel ma postać on(X, Y), wtedy:

$$L = (\Sigma \text{ wszystkich klocków na klocku X w opisie stanu bieżącego} * 2) + (\Sigma \text{ wszystkich klocków na klocku Y w opisie stanu bieżącego} * 2) + 2$$

- jeśli wybrany podcel jest już spełniony w opisie stanu docelowego, wtedy L = 0.

5. Złożoność algorytmu zachłannego

Przyjmując, że n jest liczbą wszystkich klocków w zadaniu, zawsze istnieje n podcelów do uszeregowania (patrz graf ograniczeń kolejnościowych). Łatwo można zauważyć, że maksymalna liczba decyzji o wyborze kolejnego podcelu do realizacji jest równa n-1. Ponadto, w celu ustalenia liczby L, algorytm musi w każdym kroku zliczyć wszystkie klocki występujące w zadaniu. Przyjmując, że k jest liczbą wież występujących w stanie docelowym, w każdym kroku istnieje k podceli do wyboru. Sumując, maksymalna liczba operacji do wykonania przez algorytm zachłanny jest równa:

$$N = k * n * (n-1)$$

Ostatecznie, ponieważ $k \leq n$, maksymalny czas pracy algorytmu zachłannego T_{max} wynosi:

$$T_{max} = O(n^3)$$

Wynika z tego, że algorytm zachłanny w „świecie klocków" jest wielomianowo złożony z czasie.

6. Dyskusja - optymalność algorytmu zachłannego w „świecie klocków"

Algorytmy zachłanne generują rozwiązania optymalne dla specjalnej klasy problemów. Problemy te charakteryzują się właściwością, zwaną właściwością wyboru zachłannego [2]: globalne optymalne rozwiązanie problemu jest funkcją lokalnych optymalnych rozwiązań. Właściwość ta w zasadniczy sposób odróżnia strategię zachłanną od programowania dynamicznego, w którym decyzje są także podejmowane w każdym kroku, natomiast właściwy wybór zależy również od rozwiązań podproblemów. Wybory podejmowane w algorytmie zachłannym mogą zależeć od dotychczasowych decyzji, lecz nie mogą być uzależnione od przyszłych wyborów.

Nie istnieje ogólna metoda służąca do wykazywania optymalności strategii zachłannych [2]. W niektórych problemach można posłużyć się indukcją matematyczną, w innych pomocna wydaje się teoria matroidów.

Przyjmijmy, że zadanie ze „świata klocków" reprezentowane jest przez nieskierowany graf G = (V, E), gdzie V jest zbiorem wierzchołków, natomiast E - zbiorem krawędzi. Każdy możliwy stan świata wynikający z grafu ograniczeń kolejnościowych jest reprezentowany przez wierzchołek, natomiast zbiory operatorów pomiędzy stanami są reprezentowane przez krawędzie.

Opierając się na takim grafie można zbudować tzw. matroid grafowy $M_G = (S_G, \phi_G)$ według następujących reguł:

- $S_G = E$ (zbiór krawędzi);
- jeśli A jest podzbiorem E, wtedy $A \in \phi_G$ wtedy i tylko wtedy, jeśli A jest acykliczny.

Istnieją również twierdzenia [2], które można by wykorzystać przy wykazywaniu poprawności algorytmu zachłannego w „świecie klocków":

Twierdzenie 1: Jeśli G jest grafem nieskierowanym, wtedy $M_G = (S_G, \phi_G)$ jest matroidem.

Twierdzenie 2. Matroidy posiadają właściwość wyboru zachłannego.

Warunkiem koniecznym optymalności algorytmu zachłannego jest, aby graf stanów „świata klocków" był grafem nieskierowanym. Warunek ten jest spełniony, ponieważ po przejściu do pewnego stanu „świata kocków" w zadaniu zawsze można powrócić do stanu poprzedniego. Pokazując następnie, że podzbiór operatorów przekształcających stan początkowy zadania w stan docelowy tworzy graf acykliczny, można by spełnić warunki wystarczające optymalności strategii zachłannej.

Niniejsza praca była finansowana z funduszu badań własnych BW 406/Rau-1/98/6.

LITERATURA

1. Garey M.R., Johnson D.S., Computers and Intractability - A Guide to the theory of NP-Completeness, Freeman, San Francisco, CA, 1979.
2. Cormen T.H., Leiserson Ch.E., Rivest R.L.: Introduction to Algorithms. The Massachusetts Institute of Technology, 1994.
3. Nebel B., Koehler J.: Plan reuse versus plan generation: a theoretical and empirical results. Artificial Intelligence 76 (1995) pp. 427-454.
4. Nilson N. J.: Principles of Artificial Intelligence. Tioga Publishing Company, Palo Alto1980, California.
5. Joslin D., Roach J.: A Theoretical Analysis of Conjuctive-Goal Problems. Artificial Intelligence 41 (1989/90) pp. 97-106.
6. Gałuszka A.: Block World - State Space Searching, Proc. IMACS World Congress'97, Berlin, 1997, vol.6, pp. 571-576.
7. McDermott D., Hendler J.: Planning: What it is, What it could be, An introduction to the Special Issue on Planning and Scheduling. Artificial Intelligence 76 (1995) pp. 1-16.

8. Sierocki I.: A Serial Decomposition of Planning Problems. Proc. Fourth International Symposium on Methods and Models in Automation and Robotics. Międzyzdroje, Poland, 1997, pp. 1179-1184.
9. Barret A., Weld D.S.: Partial-Order planning: evaluating possible efficiency gains. Artificial Intelligence 67 (1994) pp. 71-112.

Recenzent: Prof.dr h.inż. Jerzy Klamka

Abstract

Block world is a convenient representation for planning problems. There are many different algorithms of state space searching which use STRIPS representation for planning. Some of them search for solution through a space of world-states. In general to generate an optimal plan for block world instances, where the goal is a complete description of a set of stacks, is NP- hard.

In the paper a greedy strategy used to serialize instance subgoals is presented. It is shown when such strategy can reduce generate optimal plan problem to polynomial time problem and still generates optimal solution.

An example explain greedy algorithm in block world. The problem of greedy algorithms optimality is also discussed.