STRIPS REPRESENTATION AND NON-COOPERATIVE STRATEGIES IN MULTI-ROBOT PLANNING

Adam Gałuszka and Andrzej Świerniak
Institute of Automatic Control
Silesian University of Technology
Akademicka 16, 44-100 Gliwice, POLAND
E-mail: agaluszka@ia.polsl.gliwice.pl

KEYWORDS
Planning problems, multi-robot environment, STRIPS system, complexity of planning, non-cooperative strategies

ABSTRACT

In multi-agent (multi-robot) environment each agent tries to reach its own goal and it implies that in most cases the agent goals conflict. Under some assumptions such problems can be modelled as a STRIPS system (for instance Block World environment) with one initial state and alternative of goal states. If STRIPS planning problem is invertible then it is possible to apply machinery for planning in the presence of incomplete information to solve the inverted problem and then to find a solution for the original problem. In the paper we propose the planning algorithm that solves problem described above and, based on known results, we analyse its computational complexity.

INTRODUCTION

In multi-agent (multi-robot) environment each agent tries to achieve its own goal (Boutilier and Brafman 2001, Kraus et al. 1998). It leads to complications in problem modelling and searching for solution: in most cases agent goals are conflicting, agents have usually different capabilities and goal preferences, agents interact with problem environment simultaneously.

In this research problem environment was modelled as Block World with STRIPS representation. This domain is often used to model planning problems (Boutilier and Brafman 2001, Kraus et al. 1998, Smith and Weld 1998, Galuszka and Swierniak 2001) because of complex actions definition and simple physical interpretation. Starting from 1970s STRIPS formalism (Nilson 1980) seems to be the most popular for planning problems (Weld 1999). Planning problems algorithms usually are at least NP- hard, even in Block World environment (here the problem of optimal planning is NP-complete).

Block World today is stated an experimentation benchmark for planning algorithms (Howe and Dahlman 2002). Also more realistic situations can be presented as Block World problems, where moving blocks correspond to moving different objects like packages, trucks and planes (Slaney and Thiebaux 2001). The case of Block World problem where the table has a limited capacity corresponds to a container loading problem (Slavin 1996).

PROBLEM DEFINITION

We focus on the following situation:

- in the initial state there are a finite number of blocks and a table with unlimited place;
- two (or, in general case, more) robots want to rebuilt the initial state, each in its own way (each robot wants to achieve its own desired goal situation);
- goal of each robot consists of subgoals;
- each subgoal has its preference (subgoals are more or less important for robots);
- robots have different capabilities (i.e. each robot is not able to move all blocks);
- robots can not cooperate (this assumption is justified in the case where in the environment the communication is not allowed or communication equipement is broken down).

We are interested in the following two problems:

- to find a solution for above situation;
- to analyse computational complexity of searching for this solution.

METHOD OF FINDING A SOLUTION

The problem where there are some possible initial states and one goal state is called problem of planning in the presence of incompleteness. The inverted problem is the situation with one initial state and more possible goal states. It corresponds to multi-robot Block World problem where each robot wants to achieve its own goal. If we are able to find a plan for problem of planning in the presence

Proceedings 15th European Simulation Symposium
Alexander Verbraeck, Vlatka Hlupic (Eds.)

ISBN 3-936150-28-1 (book) / 3-936150-29-X (CD)

of incompleteness, then it is possible to extract solution for multi-agent problem.

Below we define STRIPS System, invertible planning problem and inverse operators.

Strips system

In general, STRIPS system is represented by four lists (*C; O; I; G*) (Bylander 1994, Nilson 1980):
- a finite set of ground atomic formulas (*C*), called conditions;
- a finite set of operators (*O*);
- a finite set of predicates that denotes initial state (*I*);
- a finite set of predicates that denotes goal state (*G*).

Initial state describes physical configuration of the blocks. Description should be complete i.e. should deal with every true predicate corresponding to the state. Goal state is a conjunction of predicates. In multi-agent environment each agent defines own goal. This description does not need to be complete. The algorithm results in an ordered set of operators which transforms an initial state into a goal state.

Operators *O* in STRIPS representation consist of three sublists: a precondition list (*pre*), a delete list (*add*) and an add list (*del*). Formally an operator $o \in O$ takes the form *pre(o)* $\rightarrow$ *add(o), del(o*). The precondition list is a set of predicates that must be satisfied in world-state to perform this operator. The delete list is a set of predicates that stay false after performing the operator and the add list is a set that stay true. Two last lists show effects of operator performing in problem state. Following (Koehler and Hoffmann 2000) the set of actions in a plan is denoted by P^O.

It is assumed that agents can have different capabilities (i.e. can deal with limited problem elements) and no negotiations are allowed. No negotiation assumption is satisfied in all situations where communication between agents is not allowed by problem environment or communication system fails. The case with negotiation is described for instance in (Kraus et al. 1998).

Goal preferences are also considered. We will understand the profit as a sum of preferences of goals being satisfied.

Invertible Planning Problem
Definition of Invertible Planning Problem (Koehler and Hoffmann 2000) The problem (*C, O, I, G*) is called *invertible* if and only if

$$\forall s : \forall P^O : \exists \overline{P}^O : \mathrm{Re}\,sult(\mathrm{Re}\,sult(s, P^O), \overline{P}^O) = s,$$

where

Result(*S*, <>) = *S*,
Result(*S*, <o>) = (*S* $\cup$ *add(o)*) \ *del(o)* if *pre(o)* $\subseteq$ *S* ,
S in the opposite case,
Result(*S*, <o_1, o_2,....., o_n>) = *Result*(*Result*(*S*, <o_1>), <o_2,...., o_n>),

and $\overline{P}^O$ is called *an inverted plan*.

Inverse Operator
Definition of Inverse Operator (Koehler and Hoffmann 2000). An operator $\overline{o} \in O$ is called inverse if and only if it has the form $pre(\overline{o}) \rightarrow add(\overline{o}), del(\overline{o})$ and satisfies the conditions:

1. $pre(\overline{o}) \subseteq pre(o) \cup add(o) \setminus del(o)$
2. $add(\overline{o}) = del(o)$
3. $del(\overline{o}) = add(o)$.

Under closed world assumption condition applying an inverse operator leads back to previous state. It is proved that if there is an inverse operator for each operator, then the problem is invertible.

There are assumed four classical operators in Block World (Nilson 1980). The only difference is that operators *stack* and *unstack* precise only the block that is currently transformed (i.e. do not precise on which block is stacked a transformed blocks and from which block is unstacked a transformed block):

- pickup(x) - block x is picked up from the table;
  precondition list & delete list:
  ontable(x), clear(x), handempty
  add list: holding(x)
- putdown(x) - block x is put down on the table;
  precondition list & delete list: holding(x)
  add list: ontable(x), clear(x), handempty
- stack(x) - block x is stacked on any block y;
  precondition list & delete list:
  holding(x), clear(y)
  add list: handempty, on(x,y), clear(x)
- unstack(x) - block x is unstacked from any block y;
  precondition list & delete list:
  handempty, clear(x), on(x,y)
  add list: holding(x), clear(y).

It is easy to see that *unstack* is an inverse operator for *stack* and *pickup* is an inverse operator for *putdown*. We have defined Block World as an invertible planning problem because it allows to apply planning in the presence of incompleteness methodology to search for solution of inverted multi-agent problem and then to extract solution for the right multi-agent problem.

Plan in the presence of incompleteness as an inverted plan in multi-robot environment

Algorithm of planning in the presence of incompleteness handle planning problems with uncertainty in the initial state (e.g. Weld et al. 1998). In this case the algorithm seeks to generate a robust plan by thinking over all possibilities. This approach is called *Conformant planning* (Smith and Weld 1998). Conformant planning algorithms develop non-conditional plans that do not rely on sensory information, but still succeed no matter which of the allowed states the world is actually in.

Simulation results

Block world environment was implemented using PDDL language (Planning Domain Definition Language) extended for handling uncertainty in the initial state (Yale Center… 1998). Sensory Graphplan algorithm was used to solve block world problems with uncertainty in the initial state (www.cs.washington.edu/research/projects/www/sgp.html). Two different problems are presented below. In both cases 2 robots are operating in the environment. In problem 1 Robot 1 is capable of moving blocks A,B and C whereas robot 2 can move blocks D, E and F. In Problem 2 Robot 1 is capable of moving blocks A, B, C and D whereas robot 2 can move blocks E, F, G and H. In both cases definitions of the operators are inverted (operator names are changed i.e. *unstack* for *stack* and *pickup* for *putdown*). It implies that the plan for the inverted problem is extract just by executing founded plan in the inverted order. In both cases agents goals are in conflict. The case when in multi-agent environment the goals do not conflict was explored in (Galuszka and Swierniak 2002).

Problem 1. The initial state is presented on figure 1. The goal state of robot 1 is on figure 2 and the goal state of robot 2 is on figure 3.

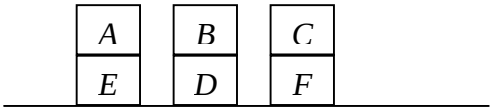

Figure 1: Initial state

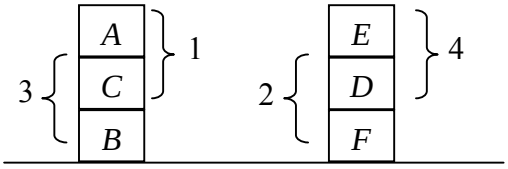

Figure 2: Desired goal state of robot 1 (the goal conflicts with the goal of robot 2) (each goal has its preference)

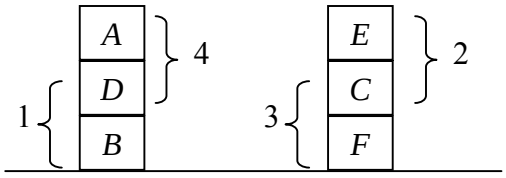

Figure 3: Desired goal state of robot 2 (the goal conflicts with the goal of robot 1) (each goal has its preference)

Solution to two-robot problem 1 (steps from 1 to 7):
2 contexts
step 7 - ((((STACK2 E)))
step 6 - (((PICK-UP2 E)) ((STACK1 A)))
step 5 - (((STACK2 D)) ((UNSTACK1 A)))
step 4 - (((PICK-UP2 D)) ((STACK1 C)))
step 3 - (((PUT-DOWN2 D)) ((UNSTACK1 C)))
step 2 - (((PICK-UP2 D)) ((PUT-DOWN1 B)))
step 1 - (((UNSTACK1 B))))

Problem 2. The initial state is presented on figure 4. The goal state of robot 1 is on figure 5 and the goal state of robot 2 is on figure 6.

Figure 4. Initial state for problem 2

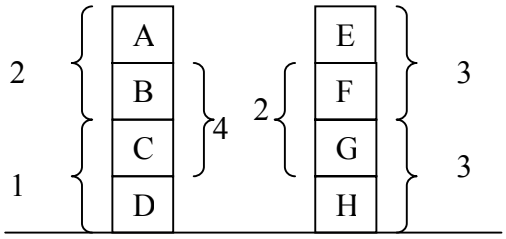

Figure 5: Desired goal state of robot 1 (the goal conflicts with the goal of robot 2) (each goal has its preference)

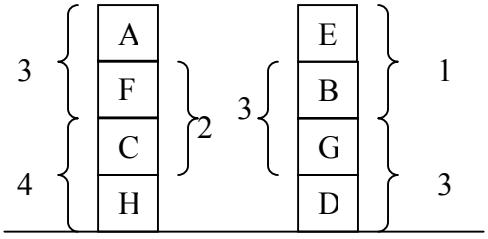

Figure 6: Desired goal state of robot 2 (the goal conflicts with the goal of robot 1) (each goal has its preference)

Solution for this two-robot problem 2 (steps from 1 to 6):

2 contexts
step 6 - ((((STACK2 E)) ((STACK1 A)))
step 5 - (((PICK-UP2 E)) ((PICK-UP1 A)))
step 4 - (((STACK2 F)) ((STACK1 B)))
step 3 - (((PICK-UP2 F)) ((PICK-UP1 B)))
step 2 - (((STACK2 G)) ((STACK1 C)))
step 1 - (((PICK-UP2 G)) ((PICK-UP1 C))))

Both agents can apply the above-founded plan to satisfy their goals. However, when they are trying to achieve their goals simultaneously they are in conflict. Now we define the non-cooperative equilibrium (Nash equilibrium) [6] and indicate how the agents can maximise their profits (the sum of preferences of satisfied goals) by achieving non-cooperative (Nash) equilibrium.

Non-cooperative equilibrium strategy

For presented problem a plan exists only if operators *stack* and *unstack* have only 1 parameter so they do not precise from which and on which block is stacked or stacked out. It implies that both agents to reach theirs goals can apply the founded plan but not simultaneously. When the goals preferences are also considered then it is possible to use Nash equilibrium strategy to precise how to apply the plan simultaneously and maximise the profit (the sum of satisfied goals preferences). The analysis of the problem leads to two remarks:

Remark 1. It is not always possible to find Nash strategy for defined problems and in general case it is depended on size of the problem.

Remark 2. More precisely the Nash strategy (if exists) defines the equilibrium for the whole plan when the number of *stack* operators in founded plan is even for each agent (2 operators for each agent in problem 1). When this condition is not satisfied (3 operators for each agent in problem 2) then the Nash strategy defines equilibrium only for a part of the problem.

The conflict between agents will be presented by a bimatrix game. Matrix A characterises the costs of the first agent (the profit with the negative sign), matrix B characterises the wastage of the second agent. We assume that agent 1 chooses rows and agent 2 chooses columns of the matrices. The agents are trying to minimise cost functions defined by matrices $A = \{a_{ij}\}$ and $B = \{b_{ij}\}$.

Definition of Nash equilibrium. The strategy $\{i_0, j_0\}$ determines non-cooperative (Nash) equilibrium in bimatrix game (A,B) if the following inequalities are satisfied:

$$a_{i_o j_o} \le a_{i j_o}$$

$$b_{i_o j_o} \le b_{i_o j}$$

for all $i = 1, 2... n,\ j = 1, 2 ... m.$

Now we define the matrixes for problem 1. The strategies in matrices are corresponding to the plan that solves the problem 2. Agent 1 can stack block C either on B or F and block A on C or D whereas agent 2 can stack block D on B or F and block E on C or D. Values in matrices correspond to goal preferences (e.g. robot 1 stacks block A on C and robot 2 block D on F then profit of robot 1 is 5 – it satisfied 2 its subgoals - whereas profit of robot 2 is 0 – it satisfied none of its subgoals).

Table 1: Matrix A (profits of the first agent)

1 \ 2	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

Table 2: Matrix B (profits of the second agent)

1 \ 2	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

Table 3: Matrix A (costs of the first agent)

1 \ 2	stack D B	stack D F	stack E C	stack E D
stack C B	- 3	- 5	(- 3)	- 7
stack C F	0	- 2	0	- 4
stack A C	- 1	- 3	- 1	- 5
stack A D	0	- 2	0	- 4

Table 4: Matrix B (costs of the second agent)

1 \ 2	stack D B	stack D F	stack E C	stack E D
stack C B	- 1	0	(- 2)	0
stack C F	- 4	- 3	- 5	- 3
stack A C	- 1	0	- 2	0
stack A D	- 5	- 4	- 6	- 4

In this game we found one strategy that satisfies non-cooperative (Nash) equilibrium definition (in brackets). This strategy modifies the plan in such a way that agent 1 should place block C on B and agent 2 should place block

E on C. It leads to the situation when the final state for the problem 2 takes the form (figure 7).

Figure 7: Final state for problem 2 comes from Nash equilibrium

Finally, the profit of the first agent is now 3 + 2 = 5 and for the second agent 2 + 4 = 6.

COMPUTATIONAL COMPLEXITY OF SEARCHING FOR THE SOLUTION

In general planning with complete information is *NP*-complete. Planning in the presence of incompleteness belongs to the next level in the hierarchy of completeness (Baral at al. 2000). A condition for 'incomplete' block world problems that reduced complexity of finding a plan to the class *P* will be shown.

Non-optimal planning in Block World is easy (Gupta and Nau 1992, Bylander 1994). To analyse complexity in our case it is assumed that planning problems are limited to only completely decomposed initial stae (i.e. all blocks are on the table) as it is shown in the example 2. Then the inverted problem is to decompose all possible initial states (i.e. goal definition consists only of 'on-table' predicates). The number of possible initials corresponds to number of robots. So the inverted problem is planning in the presence of incompleteness.

Now it wil be shown a class of 'incomplete' block world problems for which finding a plan is also easy. In this class each block has the same position in stack in each possible initial state. This class belongs to the same class of complexity as classical block world planning.

To represent possible initial states of block world it will be used Hass Diagram (Gupta and Nau 1992). This diagram is a directed acyclic graph whose nodes are the blocks and arcs are from block x to y if and only if on(x,y) is in initial state. This diagram can be constructed in linear time (Gupta and Nau 1992). Since the number of possible initial states is bounded by number of robots then time necessary to built Hass diagram for all initial states is also linear.

Next for each block in each possible initial state the position in stack is calculated using Hass diagrams. It corresponds to the problem of lenght of path in a acyclic graph. In the problem 2 block positions are:

- for A and E – 3,
- for F and B – 2,
- for C and G – 1,
- for D and H – 0.

for two possible initial states.

If each block has the same position in stack in each possible initial state then there exists the same plan for each agent that solves the problem of decomposing all initials. So to find a plan only goal situation and block positions can be considered. Subgoals are serialised in decreasing order according to block positions. Then each subgoal can be solved using only 1 macro-operator. In Example 2 we have an order:

{(on-table A), (on-table E), (on-table F), (on-table B), (on-table C), (on-table G), (on-table D), (on-table H)}

Such order leads to a solution of Example 2.

Each step requires only polynomial-time, so presented planning problem is solved in polynomial-time (belongs to class *P* of complexity).

CONCLUSION

Defining Block World environment as an invertible STRIPS planning problem allows to apply planning in the presence of incompleteness as a machinery of searching for a solution of inverted multi-agent problem and then extraction of a solution for the primary multi-agent problem. It is possible to use non-cooperative equilibrium strategy to improve the founded plan.

The result obtained for problems 1 and 2 using non-cooperative equilibrium definition should be understood as only example how to apply game-theoretic approach to solve rather narrow class of planning problems. The wide class of problems were not shown here e.g. how to modify the plan when there are more than one Nash equilibrium point, how to extend this methodology for more agents.

More solved problems can be found on www.zts.ia.polsl.gliwice.pl/galuszka/index1.htm.

Acknowledgement
This work was supported by *State Committee for Scientific Research* grant No. 4 T11A 012 23 for the year 2003.

REFERENCES

Baral Ch., V. Kreinovich, R.Trejo. 2000. Computational complexity of planning and approximate planning in

the presence of incompleteness. *Artificial Intelligence*, 122: 241-267.

Boutilier C., Brafman R.I. 2001. Partial-Order Planning with Concurrent Interacting Actions. *Journal of Artificial Intelligence Research,* 14:105-136.

Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence*, 69:165-204.

Gałuszka A, A. Świerniak. 2002. 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, pp.58-63.

Gupta N., D.S. Nau. 1992. On the complexity of Blocks-World Planning. Artificial Intelligence, 56(2-3):223-254.

Howe A.E., E.Dahlman. 2002. A Critical Assesment of Benchmark Comparison in Planning. *Journal of Artificial Intelligence Research* 17 (2002), pp. 1-33.

Koehler, J.; J. Hoffmann. 2000. On Reasonable and Forced Goal Orderings and their Use in an Agenda-Driven Planning Algorithm. *Journal of Artificial Intelligence Research*, 12 (2000), pp. 339–386.

Kraus, S.; K. Sycara; A. Evenchik. 1998. Reaching agreements through argumentation: a logical model and implementation. *Artificial Intelligence,* 104:1-69.

Mesterton-Gibbons, M. 2001. *An Introduction to Game-Theoretic Modelling*. American Mathematical Society.

Nilson, N.J. 1980. *Principles of Artificial Intelligence.* Toga Publishing Company, Palo Alto, 1980.

CA.Slaney J., S. Thiebaux. 2001. Block World revisited. *Artificial Intelligence* 125 (2001) 119-153.

Slavin T. 1996. Virtual port of call. *New Scientist*, June 1996, pp. 40-43.

Smith, D.E.; D.S. Weld. 1998. Conformant Graphplan. *Proc. 15th National Conf. on AI*.

Weld, D.S. 1999. Recent Advantages in AI Planning. Technical Report UW-CSE-98-10-01, *AI Magazine, 1999.*

Weld, D.S., C.R. Anderson i D.E. Smith. 1998. „Extending Graphplan to Handle Uncertainty & Sensing Actions". *Proc. 15th National Conf. on AI, 897-904*.

Yale Center for Computational Vision and Control. 1998, *PDDL – The Planning Domain Definition Language*, Tech Report CVC TR-98-003/DCS TR-1165.

AUTHOR BIOGRAPHIES

ANDRZEJ ŚWIERNIAK received M.Sc., Ph.D. and D.Sc. (habilitation) degrees in control engineering respectively in 1972, 1978 and 1988 all from the Department of Automatic Control, Silesian University of Technology in Gliwice, and M.A. in mathematics in 1975 from University of Silesia in Katowice, Poland. He is currently a professor at the Silesian University of Technology. His research interests are in modern control and optimisation theory, biomedical modelling and control, artificial intelligence and CADM.

ADAM GAŁUSZKA was born Ruda Slaska, Poland in June 6, 1972. He received M.Sc. degree in automation and robotics Silesian University of Technology, Poland, in 1996. Since 1996 he has been a doctorate student at the Silesian University of Technology and Teaching Assistant in Dept. of Automatic Control. In 2001 he received Ph.D. degree in automation and robotics from Silesian University of Technology, Poland. He is interested in artificial intelligence planning algorithms with STRIPS representation.