Invertible Planning and Non-Cooperative Equilibrium Strategies in Multi-Agent Planning

Adam Galuszka, Andrzej Swierniak

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. However, there exists a group of problems where it is possible to find a way to satisfy all goals even in conflicting situations. Such problems can be modelled as a STRIPS system (for instance Block World environment). If STRIPS planning problem is invertible then it is possible to apply planning under uncertainty machinery to solve inverted problem and then find a plan that solves multi-agent problem. In the paper multi-agent Block World environment as an invertible STRIPS system is presented. Two cases when agents goals conflict are explored. When agents goals conflict and agents have different goal preferences we show that it is possible to use noncooperative equilibrium strategy for modification of founded plan. This modification guarantees the best solution (in the sense of non-cooperative equilibrium) for all agents or only improves the plan.**

Index Terms-- **artificial intelligence**, **multi-agent environment, STRIPS system, invertible planning problems, non-cooperative equilibrium**

I. 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 conflict, agents have usually different capabilities and goals preferences, agents interact with problem environment simultaneously.

In our case 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 NP- hard, even in Block World environment.

Block World today stated as an experimentation benchmark for planning algorithms (Howe and Dahlman 2002). Also more realistic situations can be presented as Block World problems, where moving blocks corresponds 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 container loading problem (Slavin 1996).

A. Problem definition

In general, STRIPS system is represented by four lists (P; O; I; G) (Bylander 1994, Nilson 1980):
- a finite set of ground atomic formulas (P), called predicates;
- 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 result is an ordered set of operators which transforms an initial state into a goal state.

Operators in STRIPS representation consist of three sublists: a precondition list, a delete list and an add list. 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 satisfied goals.

Institute of Automatic Control, Silesian University of Technology
Akademicka 16, 44-100 GLIWICE, POLAND
TEL.: (+48 32) 237 27 50, FAX: (+48 32) 237 11 65

B. Invertible planning problems

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

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

Definition of Inverse Operator (Koehler and Hoffmann 2000) Let an operator $o \in O$ takes the form *pre(o)* $\rightarrow$ *add(o), del(o)*. 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 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 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 conformant planning methodology to search for solution of inverted multi-agent problem and then to extract solution for the right multi-agent problem.

C. Conformant plan as an inverted plan in multi-robot environment

Contingent planning algorithms handle planning problems with uncertainty in the initial conditions (e.g. Weld et al. 1998). In this case algorithm seek to generate a robust plan by thinking over all eventualities. 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.
The problem where there are some possible initial states and one goal state is conformant planning problem. 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 conformant planning problem then it is possible to extract solution for multi-agent problem.

II. 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 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 an 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.

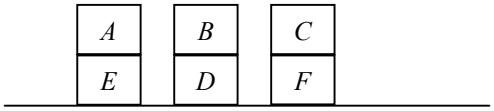

Fig.1. Initial state

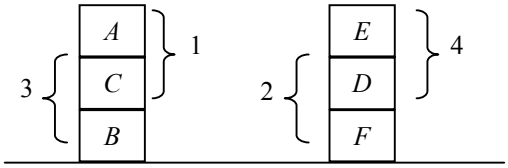

Fig.2. Desired goal state of robot 1 (goal conflicts with goal of robot 2) (each goal has its preference)

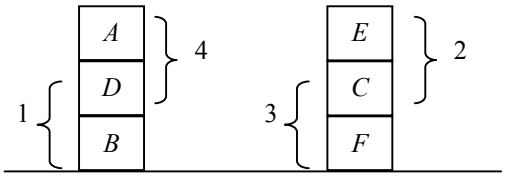

Fig.3. Desired goal state of robot 2 (goal conflicts with goal of robot 1) (each goal has its preference)

Definition of initial and goal state of the inverted problem:

```
((define (problem simple9)
   (:domain 6blocks)
  (: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 A C) (on C B)
(on E D) (on D F)
(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) ) ))
```

Solution to two-robot problem 1 (steps from1 to 7):

```
(plan 'simple7)
using backtracking csp solver
using induced mutexes
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.

Fig.4. Initial state for problem 2

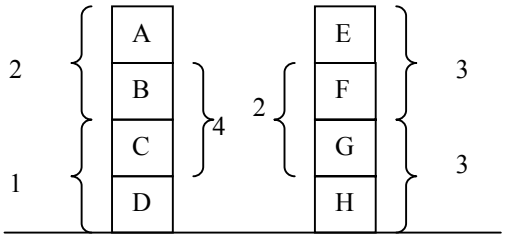

Fig.5. Desired goal state of robot 1 (goal conflicts with goal of robot 2) (each goal has its preference)

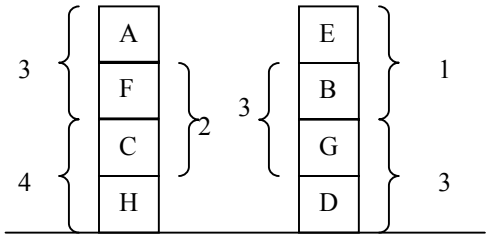

Fig.6. Desired goal state of robot 2 (goal conflicts with goal of robot 1) (each goal has its preference)

Definition of initial and goal state of the inverted problem:

```
(define (problem simple)
   (:domain 8blocks)
  (:objects A B C D E F G H)
  (:init (clear A) (clear E) (arm1-empty) (arm2-empty)
(on-table D) (on-table H)
                (or (and (on A B) (on B C) (on C D)
(on E F) (on F G) (on G H)
                         (not (on A F))  (not (on F C))
(not (on C H)) (not (on E B))
                (not (on B G)) (not (on G D))  )
                    (and (on A F) (on F C) (on C H) (on
E B) (on B G) (on G D)
                         (not (on A B)) (not (on B C))
(not (on C D)) (not (on E F))
                (not (on F G)) (not (on G H))  )
                 ) )
  (:goal (and (on-table A)(on-table B)(on-table C)(on-
table D)(on-table E)
          (on-table F) (on-table G) (on-table H) )
))
```

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

```
> (plan 'simple)
Using backtracking CSP solver
Using induced mutexes
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 to satisfy their goals can apply the above-founded plan. 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.

III. NON-COOPERAIVE (NASH) EQULIBRIUM

The conflict between agents will be presented by two-matrix 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 matrixes. The agents are trying to minimise cost functions defined by matrixes $A = \{a_{ij}\}$ and $B = \{b_{ij}\}$.

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

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

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

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

Now we define the matrixes for problem 1. The strategies in matrixes 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 matrixes correspond to goal preferences.

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

Matrix A (profits of the first 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

Matrix B (profits of the second 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

Matrix A (costs of the first 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

Matrix B (costs of the second agent)

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 (fig.7):

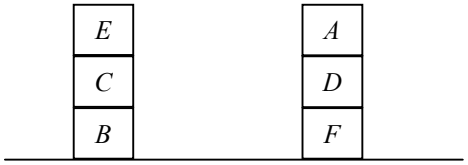

Fig.7. Final state for problem 2 comes from Nash equilibrium

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

Similarly we can define matrixes for problem 2. Costs matrixes are only presented (fig.8 and 9):

2 \ 1	stack G D	stack G H	stack F G	stack F C	stack E F	stack E B
stack C D	-1	-4	-3	-1	-4	-1
stack C H	0	-3	-2	0	-3	0
stack B C	(-4)	-7	-6	-4	-7	-4
stack B G	0	-3	-2	0	-3	0
stack A B	-2	-5	-4	-2	-5	-2
stack A F	0	-3	-2	0	-3	0

Fig.8. Matrix A (costs of the first agent)

2 \ 1	stack G D	stack G H	stack F G	stack F C	stack E F	stack E B
stack C D	-3	0	0	-2	0	-1
stack C H	-7	-4	-4	-6	-4	-5
stack B C	(-3)	0	0	-2	0	-1
stack B G	-6	-3	-3	-5	-3	-4
stack A B	-3	0	0	-2	0	-1
stack A F	-6	-3	-3	-5	-3	-4

Fig.9. Matrix B (costs of the second agent)

In this case the solution is different. We could find only 1 Nash equilibrium point (similarly like for problem 1) but now it not precise the final state of the problem. The Nash solution does not determine the placement of blocks A and E and in fact it can not since the number of blocks is too small. In this case stacking A on B implies that the second agent should stack E on F so both subgoals of the first agent are satisfied and none for the second agent. The another possibility implies that none of subgoals for the first agent are satisfied and two for the second agents. Finally the equilibrium point is reached for all blocks except A and E so Nash strategy only improved the founded plan by defining equilibrium for a part of problem elements.

IV. DISCUSSION

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.Not always is possible to find Nash strategy for defined problems and in general case it is depended of size of the problem.

Remark 2. Precisely speaking 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.

V. CONCLUSION

Defining Block World environment as an invertible STRIPS planning problem allows to apply conformant planning machinery to search for a solution of inverted multi-agent problem and then to extract a solution for the right 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 here not shown e.g. how to modify the plan when there are more than one Nash equilibrium point, how to extend this methodology for more agents.

Acknowledgement
This work was supported by KBN grant No. 4 T11A 002 24 for the year 2003.

References

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

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

[3] 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.

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

[5] 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.

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

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

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

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

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

[13] 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*.

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