Planning in multi-agent environment as inverted STRIPS planning in the presence of uncertainty

ADAM GALUSZKA, ANDRZEJ SWIERNIAK
Institute of Automatic Control
Silesian University of Technology
Akademicka 16, 44-100 Gliwice,
POLAND

Abstract: - In multi-agent (multi-robot) environment each agent tries to achieve 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 conflict situations. Such problems can be modelled as a STRIPS system (for instance Block World environment). If STRIPS planning problem is invertible than it is possible to apply planning under uncertainty methodologies to solve inverted problem and than 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 are considered and simulated: when goals conflict and do not conflict. Necessary and sufficient conditions of plan existence are formulated.

Key-Words: - multi-agent environment, STRIPS system, invertible planning problems

1 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, agent 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 complex actions definition but 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.
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).

1.1 Problem definition

Initial state describe 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 transform initial state into goal state. Operators in STRIPS representation consist of three sublists: precondition list, delete list and 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. Follow (Koehler and Hoffmann 2000) set of actions in a plan is denoted by P^O.
It is assumed agents can have different capabilities (i.e. can deal with limited problem elements) and no negotiations are allowed. Goal preferences are not considered.

2 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 $\bar{o} \in O$ is called inverse if and only if has the form $pre(\bar{o}) \rightarrow add(\bar{o}), del(\bar{o})$ and satisfies the conditions:

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

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:

- 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,y) - 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,y) - 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.

3 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.

4 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. Robot 1 is capable of moving blocks A,B and C whereas robot 2 can move blocks D, E and F. In Problem 1 goals of the robots do not conflict (Fig.1 and 2), in Problem 2 they do conflict (Fig. 3, 4 and 5). In both cases definitions of the operators are inverted (operator names are changed between *unstack* and *stack* and between *pickup* and *putdown*). It implies that plan for inverted problem is extract just by executing founded plan in inverted order.

Problem 1.

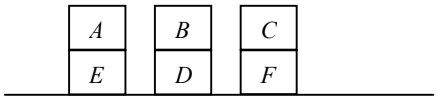
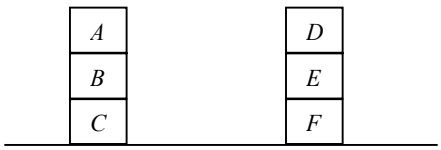


Fig.1. Initial state



a) robot 1 *b) robot 2*
Fig.2. Goal state of robots
(goals do not conflict)

Definition of the operators:

```
(define (domain galuszka5)
(:requirements :strips :equality :uncertainty :typing)
  (:predicates (on ?x ?y)
               (on-table ?x)
               (clear ?x)
               (arm1-empty)
               (arm2-empty)
               (holding1 ?x)
               (holding2 ?x))                )
  (:action put-down1
           :parameters (?ob)
          :precondition (and (not (= ?ob D)) (not (= ?ob
E)) (not (= ?ob F)) (clear ?ob) (on-table ?ob) (arm1-
empty))
              :effect
              (and (not (on-table ?ob))
                        (not (clear ?ob))
                        (not (arm1-empty))
                        (holding1 ?ob)))
  (:action pick-up1
              :parameters (?ob)
              :precondition (holding1 ?ob)
              :effect
              (and (not (holding1 ?ob))
                        (clear ?ob)
                        (arm1-empty)
                        (on-table ?ob)))
  (:action unstack1
    :parameters (?ob)
    :precondition (and (holding1 ?ob) (clear ?sunderob))
    :effect
    (and (not (holding1 ?ob))
              (not (clear ?sunderob))
              (clear ?ob)
              (arm1-empty)
              (on ?ob ?sunderob)))
  (:action stack1
    :parameters (?x)
    :precondition (and (not (= ?x D)) (not (= ?x E)) (not (=
?x F)) (clear ?x) (arm1-empty) (on ?x ?y))
    :effect
    (and (holding1 ?x) (not (clear ?x)) (not (arm1-empty))
(clear ?y) (not (on ?x ?y)) ) )
     (:action put-down2
    :parameters (?ob)
    :precondition (and (not (= ?ob A)) (not (= ?ob B))
  (not (= ?ob C)) (clear ?ob) (on-table ?ob) (arm2-empty))
    :effect
    (and (not (on-table ?ob))
     (not (clear ?ob)) (not (arm2-empty)) (holding2 ?ob)))
  (:action pick-up2
    :parameters (?ob)
    :precondition (holding2 ?ob)
    :effect
    (and (not (holding2 ?ob))
              (clear ?ob) (arm2-empty)
              (on-table ?ob)))
  (:action unstack2
    :parameters (?ob)
    :precondition (and (holding2 ?ob)
           (clear ?sunderob))
    :effect
    (and (not (holding2 ?ob))
   (not (clear ?sunderob)) (clear ?ob)
   (arm2-empty) (on ?ob ?sunderob)))
  (:action stack2
    :parameters (?x)
    :precondition (and (not (= ?x A)) (not (= ?x B))
   (not (= ?x C)) (clear ?x) (arm2-empty) (on ?x ?y))
    :effect
    (and (holding2 ?x) (not (clear ?x)) (not (arm2-empty))
(clear ?y) (not (on ?x ?y)) ) ) )
```

Definition of initial and goal state of the inverted problem:

```
(define (problem simple6)
   (:domain galuszka5)
  (:objects A B C D E F)
  (:init (clear A) (clear D) (arm1-empty)
           (arm2-empty) (on-table C) (on-table F)
           (on A B) (on B C) (on D E) (on E 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 8):

```
(plan 'simple6)
1 context
Levels
***** 1 *****Actions:  12  Propositions:  20
***** 2 *****Actions:  30  Propositions:  30
***** 3 *****Actions:  48  Propositions:  42
***** 4 *****Actions:  74  Propositions:  58
***** 5 *****Actions:  104  Propositions:  74
***** 6 *****Actions:  134  Propositions:  88
***** 7 *****Actions:  160  Propositions:  100
***** 8 *****Actions:  172  Propositions:  100

step 8 - ((( stack1  a  b)))
step 7 -  ((( stack2  d  e)) (( pick-up1  a)))
step 6 -  ((( pick-up2  d)) (( stack1  b  c)))
step 5 -  ((( stack2  e  f)) (( unstack1  b  d)))
step 4 -  ((( pick-up2  e)) (( put-down1  a)))
step 3 -  ((( put-down2  f)) (( unstack1  a  e)))
step 2 -  ((( pick-up2  f)) (( put-down1  c)))
step 1 -  ((( unstack1  c  f))))
```

Problem 2.

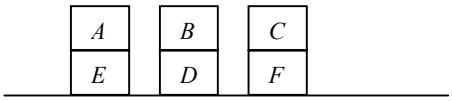


Fig.3. Initial state

Fig.4. Desired goal state of robot 1
(goal conflicts with goal of robot 2)

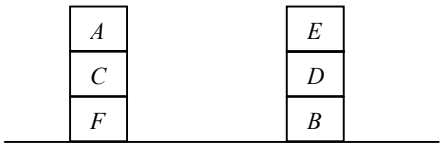


Fig.5. Desired goal state of robot 2
(goal conflicts with goal of robot 1)

Definition of the operators:

```
(define (domain galuszka6)
  (:requirements :strips :equality :uncertainty :typing )
  (:predicates (on ?x ?y)
               (on-table ?x)
               (clear ?x)
               (arm1-empty)
               (arm2-empty)
               (holding1 ?x)
               (holding2 ?x)
               )
  (:action put-down1
     :parameters (?ob)
     :precondition (and (not (= ?ob d)) (not (= ?ob e))
      (not (= ?ob f)) (clear ?ob) (on-table ?ob) (arm1-
empty))
     :effect
     (and (not (on-table ?ob))  (not (clear ?ob))
                        (not (arm1-empty)) (holding1 ?ob)))
  (:action pick-up1
     :parameters (?ob)
     :precondition (holding1 ?ob)
     :effect
     (and (not (holding1 ?ob)) (clear ?ob) (arm1-empty)
                        (on-table ?ob)))
  (:action unstack1
     :parameters (?ob)
     :precondition (and (holding1 ?ob) (clear ?sunderob))
     :effect
     (and (not (holding1 ?ob)) (not (clear ?sunderob))
     (clear ?ob) (arm1-empty) (on ?ob ?sunderob)))
  (:action stack1
     :parameters (?x)
     :precondition (and (not (= ?x d)) (not (= ?x e))
       (not (= ?x f)) (clear ?x) (arm1-empty) (on ?x ?y))
     :effect
     (and (holding1 ?x) (not (clear ?x)) (not (arm1-empty))
                        (clear ?y) (not (on ?x ?y)) ) )
  (:action put-down2
     :parameters (?ob)
     :precondition (and (not (= ?ob a)) (not (= ?ob b))
       (not (= ?ob c)) (clear ?ob) (on-table ?ob)
       (arm2-empty))
     :effect
     (and (not (on-table ?ob))  (not (clear ?ob))
    (not (arm2-empty)) (holding2 ?ob)))
  (:action pick-up2
     :parameters (?ob)
     :precondition (holding2 ?ob)
     :effect
     (and (not (holding2 ?ob)) (clear ?ob)
       (arm2-empty) (on-table ?ob)))
  (:action unstack2
     :parameters (?ob)
     :precondition (and (holding2 ?ob) (clear ?sunderob))
     :effect
     (and (not (holding2 ?ob)) (not (clear ?sunderob))
     (clear ?ob) (arm2-empty) (on ?ob ?sunderob)))
  (:action stack2
     :parameters (?x)
     :precondition (and (not (= ?x a)) (not (= ?x b))
       (not (= ?x c)) (clear ?x) (arm2-empty) (on ?x ?y))
     :effect
     (and (holding2 ?x) (not (clear ?x)) (not (arm2-empty))
                        (clear ?y) (not (on ?x ?y)) ) )    )
```

Definition of initial and goal state of the inverted problem:

```
(define (problem simple7)
   (:domain galuszka6)
  (: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 d) (on d b) (on e c) (on c f)
           (not (on a c))  (not (on e d)) )
              (and (on a c) (on c f) (on e d) (on d b)
           (not (on a d)) (not (on e c))  )          ) )
  (: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 2 (steps from1 to 5):

```
(plan 'simple7)
using backtracking csp solver
using induced mutexes
2 contexts
levels
***** 1 *****actions:  28  propositions:  24
***** 2 *****actions:  66  propositions:  38
***** 3 *****actions:  118  propositions:  58
***** 4 *****actions:  184  propositions:  78
***** 5 *****actions:  262  propositions:  94

step 5 - (((( stack2  e)))
step 4 -  ((( pick-up2  e)) (( stack1  a)))
step 3 -  ((( stack2  d)) (( unstack1  a)))
step 2 -  ((( pick-up2  d)) (( put-down1  b)))
step 1 -  ((( unstack1  b))))
```

5 Discussion

Results obtained for Problem 1 and Problem 2 should be interpreted in different ways. In Problem 1 both robots achieve their goals whereas in Problem 2 founded plan can be apply by both robots to achieve their goals but not at the same time. For problem 2 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 after applying plan in Problem 2 only one of robots achieves its goal but both have the same chances. In the case where goal preferences would be considered it influences on final state of the problem

Moreover it see that *necessary* and *sufficient* condition that plan exists is that all top and on-table

blocks of all goals of the agents have to be the same (for problem 2 both robots have clear blocks A and E, and on-table blocks F and B):

Theorem
Let n be a number of agents (robots) and G_i is a goal definition of i-agent (i=1,2,…,n) and G_j is a goal definition of j-agent (j=1,2,…,n), $j \neq i$. The *necessary* and *sufficient* condition that plan of the defined problem exists is:

$$\forall_i \quad \forall_{clear(X)\in G_i, ontable(Y)\in G_i} \quad \exists_{clear(Z)\in G_j, ontable(V)\in G_j}$$

$$X = Z, Y = V$$

Prove
Let the conditions formulated above are not satisfied for example block X is clear in goal description of agent 1 but is not clear in goal of agent 2. Then it is impossible to apply the same operator to achieve both goals until the closed world is assumed (if *clear(X)* is not mentioned in goal definition then is assumed to be false).

It should be noted that these conditions have to be satisfied only if agent goals conflict. If agent goals do not conflict then class of solvable problems is unlimited.

6 Conclusion

Defining Block World environment as an invertible STRIPS planning problem 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.
Necessary and sufficient condition that plan exists were formulated.
Cases where agent goals have different preferences were not explored.

Acknowledgement
This work has been supported by BK and KBN 8T11A01219 grants.

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] Galuszka, A. A. Swierniak. 2001. "Uncertain information in modelling and simulation of STRIPS planning problems", Proc. of European Control Conference, Porto 2001, CD-ROM.

[4] 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.

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

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

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

[8] Weld, D.S. 1999. "Recent Advantages in AI Planning." Technical Report UW-CSE-98-10-01; to appear in *AI Magazine, 1999.*

[9] 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*.

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

Appendix

Simulation results for 2 agents and 8 blocks (robot 1 is capable to move blocks A,B,C,D, robot 2 – blocks E,F,G,H).

Definition of initial and goal state of the inverted problem:

```
(define (problem simple8)
   (:domain galuszka7)
  (:objects A B C D E F G H)
  (:init (clear A) (clear B) (clear C) (arm1-empty) (arm2-
empty)
 (on-table D) (on-table E) (on-table F)
(or (and (on A G) (on G D) (on B H) (on H E) (on C F)
(not (on A H))  (not (on B G))  )
                        (and (on A H) (on H E) (on B G)
                                (on G D) (on C F)
(not (on A G))   (not (on B H)) )) )
  (:goal (and (on A B) (on C D) (on E F) (on H G) (on-
table B) (on-table D)(on-table F) (on-table G) ) ))
```

Solution to the problem (steps from1 to 8):

```
(plan 'simple8)
2 contexts
Levels
***** 1 *****Actions:  36  Propositions:  30
***** 2 *****Actions:  92  Propositions:  54
***** 3 *****Actions:  192  Propositions:  85
***** 4 *****Actions:  320  Propositions:  119
***** 5 *****Actions:  460  Propositions:  149
***** 6 *****Actions:  552  Propositions:  163
***** 7 *****Actions:  582  Propositions:  164
***** 8 *****Actions:  584  Propositions:  164
***** 9 *****Actions:  584  Propositions:  164
step 8- (((( STACK1  B))) ((( PICK-UP1  B)))
step 7- ((( STACK1  A)))
step 6- ((( STACK1  A)) (( STACK2  G)))
step 5- ((( PICK-UP2  G)) (( UNSTACK1  A)))
step 4- ((( STACK1  C)) (( STACK2  H)))
step 3- ((( UNSTACK2  H)) (( UNSTACK1  C)))
step 2- ((( STACK1  C)) (( PUT-DOWN2  E)))
step 1- ((( UNSTACK2  E)) (( UNSTACK1  C))))
```