Scoring Functions of Approximation of STRIPS Planning by Linear Programming – Block World example

Adam Galuszka
Silesian University of Technology,
Akademicka 16, 44-100 Gliwice, Poland,
{Adam.Galuszka@polsl.pl}

Abstract :- STRIPS planning is a problem of finding of a set of actions that transform given initial state to desired goal situation. It is hard computational problem. In this work an approximation of STRIPS block world planning by linear programming is shown. The cost of such approach is that algorithm can results in non-interpretable solutions for some initial states (what is followed by assumption P ≠ NP). This is because the discrete domain (truth or false) is transformed to continuous domain (LP program). Additionally two scoring functions have been introduced to estimate quality of the plan. Proposed approach is illustrated by exemplary simulation.

Keywords :- STRIPS planning, Block world, computational efficiency, linear programming.

1 Introduction

Artificial Intelligence is a study of design of intelligent agents. An intelligent agent is a system that acts intelligently on its environment. There are various problems which are being investigated by Artificial Intelligence, like knowledge, reasoning, learning and planning (Weld 1999, Chen et al. 2008). Planning is a task of coming up with a sequence of actions that will achieve a goal. Finding an optimal plan is generally a hard computational problem and needs a lot of resources.

Planning should be distinguished from *scheduling* – well-known and frequently used technique of improving the cost of a plan. Planning is understood as causal relations between actions, while scheduling is concerned with metric constraints on actions (Backstrom 1998). When all states of a planning problem (including an initial and a goal) are described by a given set of conditions (also called predicates), then the problem is called STRIPS planning problem (Nilson 1980). There are many applications of the planning problems in industrial processes, production planning, logistics and robotics (Backstrom 1998, Karadimas et al. 2008). The STRIPS system has been successfully applied in planning modules of Deep Space One Spacecraft (Weld 1999) and for elevators control in Rockefeler Center in New York (Koehler and Schuster 2000).

In this case, the planning problem environment is modeled as a Block World using the STRIPS representation. This domain is often used to model planning problems (Slaney and Thiebaux 2001, Boutilier and Brafman 2001, Kraus et al 1998, Smith and Weld 1998, Rintanen 1999, Galuszka and Swierniak 2004, Sierocki 1997) because of complex action definition and simple physical interpretation. One can say that starting from 1970s, STRIPS formalism (Nilson 1980) seems to be very popular for planning problems (Weld 1999). Today this domain can be a representation for logistic problems, where moving blocks corresponds to moving different objects like packages, trucks and planes (e.g. 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). In real situation decision problems at container terminals are more complex and divided into several groups: arrival of a ship, unloading and loading of a ship, transport of containers from and on a ship, stacking of containers (see e.g. www.ikj.nl/container/decisions.html). Since arrival of a ship and containers transport are usually treated as scheduling and allocation problems (e.g. Imai et al. 2001, Bish et al. 2001), problems of loading and unloading and container stacking can be treated as planning problems (e.g. Wilson 2000). In natural way containers can be treated as blocks and cranes as robots that are stacking and unstacking blocks. In these cases it is shown that planning algorithm in practice has to be efficient in time and generate high quality plans (close to optimal).

To increase computational efficiency of finding a plan one can transform planning problem to another (simpler) problem and search for the solution of the transformed problem. To increase computational efficiency of planning a transformation to Linear Programming problem is shown. The idea of representing STRIPS planning problems by linear constraints and objective function is not new in the literature (see e.g. Nareyek et al. 2005). In these cases the planning problem takes the form of binary integer linear program. It implies that the only allowed values of variables are '0' and '1' and they corresponds to false/truth values of planning problem predicates and actions. The computational efficiency of the approach is low (because of complexity of integer programming algorithms) and solution can be found only for small size planning problems. That is why it is proposed here to analyze the solution of the linear relaxation of binary integer linear program. The cost of such approach is that algorithm can results in non-interpretable solutions for some initial states (what is followed by assumption $P \neq NP$). This is because the discrete domain (truth or false) is transformed to continuous domain (LP program), so solution of LP not always can be directly interpreted as a plan. For such cases scoring functions to estimate quality of the plan and the additional heuristic are proposed in this paper.

The paper is organized as follow: STRIPS system, LP problem, and the transformation of Block World to LP program are introduced in section 2. Scoring functions are presented in section 3. Section 4 introduces an additional heuristic. In section 5 simulation results are presented. Finally, all is concluded.

2 STRIPS system, LP problem, and the transformation of Block World to LP program

In general, STRIPS language 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*).

The initial state describes the physical configuration of the blocks. This description should be complete i.e. it should deal with every true predicate corresponding to this state. The goal situation describes what should be true. Each goal consists of subgoals and has a form of conjunction of predicates. This description does not need to be complete, i.e. does not need to describe a *state* of the problem.

The algorithm results in an ordered set of operators which transforms the initial state *I* into a state with true predicates mentioned in the goal situation *G*. Operators in STRIPS representation consist of three sublists: a precondition list (*pre(o)*), a delete list (*del(o)*) and an add list (*add(o)*). The precondition list is a set of predicates that must be satisfied to apply this operator. The delete list is a set of predicates that will be false after applying the operator and the add list is a set of predicates that are true after the operator is applied. The two last lists show the effects of applying the operator into a current problem state ($S \subset C$).

Let an operator $o \in O$ takes the form $pre(o) \rightarrow add(o), del(o)$. Following (Koehler and Hoffmann 2000), the set of operators $<o_1, o_2, \ldots, o_n>$ in a plan is denoted by P^O.

If an operator $o \in O$ is applied to the current state of the problem then the state is modified. This modification is described by function *Result*:

$$Result(S, <o>) = (S \cup add(o)) \setminus del(o) \text{ if } pre(o) \subseteq S, S \text{ in the opposite case} \tag{1}$$

and

$$Result(S, <o_1, o_2, \ldots, o_n>) = Result(Result(S, <o_1>), <o_2, \ldots, o_n>). \tag{2}$$

Below four operators classical in Block World as an example of STRIPS operators are presented (Nilson 1980):

– 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).

2.1 Transformation To Linear Programming

Following (Bylander 1997) 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 the last step of planning.

As an example let us consider the problem 1 of planning in Block World environment with 4 blocks (called A, B, C, D) (Galuszka and Swierniak 2004). The goal is to decompose the initial state. It is assumed one STRIPS operator that moves the block x from the other block to the table, on(x) means that block x is on another block (or on the table), clear(x) means that there is no other block on block x :

move-to-table(x,y):
preconditions: on(x,y), clear(x)
del: on(x,y)
add: clear(y)

The goal is reached if the following conditions are true:

$$clear(A), clear(B), clear(C), clear(D) \tag{3}$$

Assume 2 steps of planning (states indexes are: 0, 1, 2). So (16+16+16) variables are needed for conditions for i = 0, 1, 2. In addition (12+12) variables are needed for operators (12 for transformation from state 0 to 1 and 12 from 1 to 2) for i = 0, 1 (see table 1). Then the value of the objective function to be maximised *F(G)* is:

F(G) = (clear(A)(2) + clear(B)(2) + clear(C)(2) + clear(D)(2)). (4)

If the goal is reached then the objective function is equal to 4 (4 conditions are true in the goal state). Constraints for Linear Programming problem are:

- at most 1 operator can be applied in each planning step:

Σ move-to-table(x)(i) = 1 (5)

– operator can not be applied unless its preconditions are true:

on(x,y)(i) ≥ move-to-table(x,y)(i) (6)

for all operators in each planning step,

clear(A)(i) ≥ move-to-table(A,B)(i)+ move-to-table(A,C)(i)+ move-to-table(A,D)(i) (7)
clear(B)(i) ≥ move-to-table(B,A)(i)+ move-to-table(B,C)(i)+ move-to-table(B,D)(i) (8)
clear(C)(i) ≥ move-to-table(C,A)(i)+ move-to-table(C,B)(i)+ move-to-table(C,D)(i) (9)
clear(D)(i) ≥ move-to-table(D,A)(i)+ move-to-table(D,B)(i)+ move-to-table(D,C)(i) (10)

for all planning steps.

Table 1. Variables of LP problem in each step

conditions		operators	
1.	*on(A,B)(i)*	1.	*move-to-table(A,B)(i)*
2.	*on(A,C)(i)*	2.	*move-to-table(A,C)(i)*
3.	*on(A,D)(i)*	3.	*move-to-table(A,D)(i)*
4.	*on(B,A)(i)*	4.	*move-to-table(B,A)(i)*
5.	*on(B,C)(i)*	5.	*move-to-table(B,C)(i)*
6.	*on(B,D)(i)*	6.	*move-to-table(B,D)(i)*
7.	*on(C,A)(i)*	7.	*move-to-table(C,A)(i)*
8.	*on(C,B)(i)*	8.	*move-to-table(C,B)(i)*
9.	*on(C,D)(i)*	9.	*move-to-table(C,D)(i)*
10.	*on(D,A)(i)*	10.	*move-to-table(D,A)(i)*
11.	*on(D,B)(i)*	11.	*move-to-table(D,B)(i)*
12.	*on(D,C)(i)*	12.	*move-to-table(D,C)(i)*
13.	*clear(A)(i)*		
14.	*clear(B)(i)*		
15.	*clear(C)(i)*		
16.	*clear(D)(i)*		

Next group of constraints describes changes of the state after applying an operator. These are equality constraints:

clear(A)(i+1) =

= clear(A)(i) + move-to-table(B,A)(i)+ move-to-table(C,A)(i)+move-to-table(D,A)(i) (11)

and similar for blocks: *B, C, D*, and:

on(A,B)(i+1) = on(A,B)(i) – move-to-table(A,B)(i). (12)

Finally constraints for variables are needed: these should be mapped between 0 and 1 values what corresponds to truth degree of the variables. Transformation to Linear Programming results to the problem with 72 variables, 34 inequality, 32 equality constraints. In each planning step there are 16 conditions and 12 possible operators.

In general case the size of LP problem is as follow (*l* denotes number of steps, *n* denotes number of blocks):

- the number of variables of LP problem:

$$(l + 1)\,(2n^2 + n), \tag{13}$$

- the number of inequality constraints:

$$l\,(n^2 + n + 1), \tag{14}$$

- the number of equality constraints:

$$l\,(n^2 + n), \tag{15}$$

so the transformation is polynomial.

If the LP problem is defined as:

$$\max_x \leftarrow f^T x \qquad \begin{aligned} Ax &\le b \\ A_{eq}x &= b_{eq} \\ 0 \le x &\le 1 \end{aligned} \tag{16}$$

then *f* defines the goal state of the planning problem, and the initial state is enclosed in A_{eq} and b_{eq}. The solution of LP problem is denoted by *xopt.*

2.1.1 Example 1

To illustrate the transformation example 1 is shown. The problem (fig.1) is to decompose the initial state defined by the set: *on(A,B), on(D,C), clear(A), clear(D).* Matrices *A* and A_{eq} that correspond to the planning problem are sparse and their nonzero elements are illustrated in the fig.2.

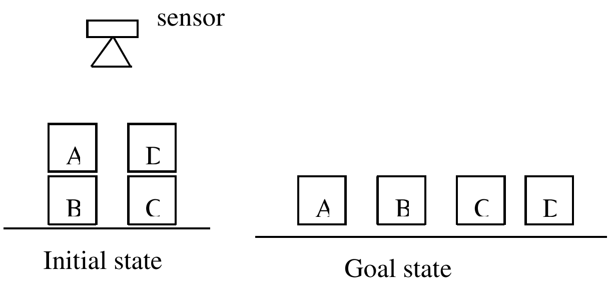

Fig.1. The initial and goal states for example 1

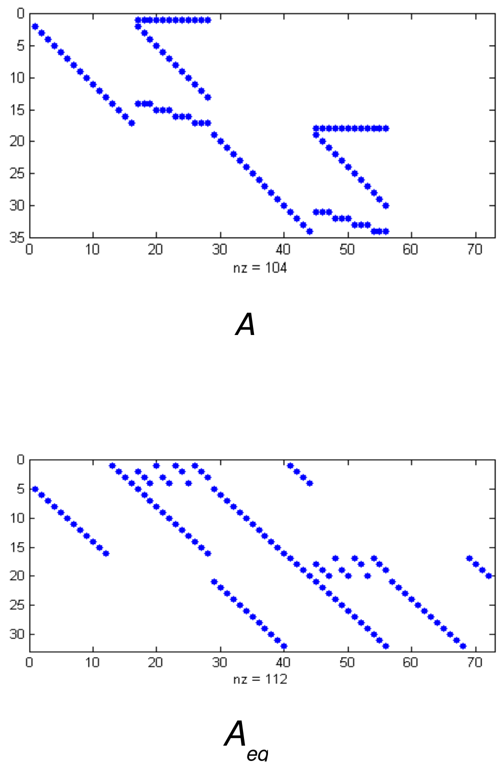

Fig.2. Matrices for linear program of example 1

The solution of Linear Programming algorithm is the set of variable values that gives optimal (maximal) value of objective function: variables from 69 to 72 are equal to 1. These corresponds to truth values of operators (variables 28 and 45): *move-to-table(D,C)(0)*=1 and *move-to-table (A,B)(1)*=1 that solve planning problem in 2 steps (Table 2).

Table 2. Solution of Example 1.

conditions *i = 0*		*actions* *i = 0*		*conditions* *i = 1*		*Actions* *i = 1*		*conditions* *i = 2*	
1.	1.0000	17.	-0.0000	29.	1.0000	45.	1.0000	57.	0.0000
2.	0.0000	18.	-0.0000	30.	0.0000	46.	0.0000	58.	-0.0000
3.	0.0000	19.	0.0000	31.	0.0000	47.	0.0000	59.	0.0000
4.	0.0000	20.	0.0000	32.	0.0000	48.	0.0000	60.	-0.0000
5.	-0.0000	21.	-0.0000	33.	-0.0000	49.	-0.0000	61.	-0.0000
6.	-0.0000	22.	0.0000	34.	0.0000	50.	0.0000	62.	0.0000
7.	0.0000	23.	0.0000	35.	-0.0000	51.	-0.0000	63.	0.0000
8.	-0.0000	24.	0.0000	36.	0.0000	52.	-0.0000	64.	-0.0000
9.	0.0000	25.	-0.0000	37.	0.0000	53.	0.0000	65.	0.0000
10.	0.0000	26.	0.0000	38.	-0.0000	54.	-0.0000	66.	0.0000
11.	-0.0000	27.	0.0000	39.	-0.0000	55.	-0.0000	67.	0.0000
12.	1.0000	28.	1.0000	40.	0.0000	56.	-0.0000	68.	0.0000
13.	1.0000			41.	1.0000			69.	1.0000
14.	-0.0000			42.	0.0000			70.	1.0000
15.	0.0000			43.	1.0000			71.	1.0000
16.	1.0000			44.	1.0000			72.	1.0000

However, this result is very optimistic: all variables (conditions) are 0's or 1's. In general case as a result we can receive any value from 0 to1. So the cost of the approach is that algorithm can results in non-interpretable solutions for some initial states (what is followed by assumption P ≠ NP). This is because the discrete domain (truth or false) is transformed to continuous domain (LP program), so solution of LP not always can be directly interpreted as a plan. For such cases scoring functions to estimate quality of the plan has been introduced in section 3, and the additional heuristic has been proposed in section 4.

3 Scoring functions

The goal of introducing scoring functions is an estimation if the LP solution is a 'perfect' solution (i.e. if it can be directly treated as the planning problem solution). The LP solution can be 'imperfect' for 2 reasons: 1) it is not binary vector (i.e. does not correspond to true/false conditions and actions), 2) the variable values that correspond to state in last step of planning does not correspond to conditions for the planning problem goal situation (i.e. the LP solution does not lead to goal of planning). Proposed scoring functions have common property: they are equal to1 if the plan is 'perfect' due to scored property, and are lower than 1 in the opposite case. The first function, *U(xopt)*, called *plan utility*, estimates how close is the LP solution to binary solution:

$$U(xopt) = \frac{l}{k} \quad , \qquad (17)$$

where: l – number of the planning steps, k – number of non-zeros variables of LP solution that correspond to actions, *xopt* – LP solution. The second function, *Sat(xopt)*, called *goal satisfaction degree*, compares the value of LP objective function to the expected value, that is equal to the number of true conditions in planning problem goal situation:

$$Sat(xopt) = \frac{f^T xopt}{m} \quad , \qquad (18)$$

where: $f^T xopt$ - the value of LP objective function, m - the number of true conditions in planning problem goal situation. This definition is based on observation that the value of the objective function can be useful as the value of the utility function of planning problem: if the value of the objective function is divided by the number of truth predicates in the goal state G it returns the 'degree' of satisfaction of the problem goal state. *Sat(xopt)* has the property that returns 1 if the plan satisfies all predicates in the goal definition, and 0 in the opposite case. For values between 0 and 1 the goal is partly satisfied.

4 Additional heuristic

In the heuristic the variables values are treated as a 'truth degrees' and the algorithm scheme can be summarized as follow:

i. Solve the Linear Programming problem that corresponds to planning problem;
ii. From the group of variables that correspond to operators in the 1st step choose the one with the biggest value of truth degree;
iii. Assume the value of this variable equal to 1 (it means that this operator is choosed as the best proposition) and add this equality as an additional constraint to the LP problem;
iv. Solve this more restricted problem;
v. Repeat steps b) to d) for the group of variables that correspond to operators in the next steps until result vector consists from only 0's and 1's.

The number of repeats is limited by the number of steps. Fortunately, most of planning problems has the property that the number of steps is polynomial limited by the size of the problem (Baral Ch., V. Kreinovich, R.Trejo. 2000). It follows that the linear transformation with additional heuristic remains polynomial. An example of 'pathologic' planning problem is Towers of Hanoi (see e.g. A. Beck, M.N. Bleicher, D. W. Crowe, Excursions into Mathematics, A K Peters, 2000): in this case the number of operators growths exponentially with the problem size. This approach does not support such group of problems.

The result of the heuristic is a binary solution of the linear program representing the planning problem, so this solution is interpreted as a plan that solves planning problem. It should be noted that the solution is feasible: in the first step of the heuristic an additional equality constraint that correspond to one action in first planning step is added (step c of the heuristic), so from constraints

(5) variables for all other actions in this step are set to 0, that corresponds to one action.

5 Simulation results

To analyze computational efficiency of transformation of presented representations of STRIPS planning the 10 block decomposition problem has been implemented and solved using MATLAB® software. The matrices of LP program are sparse (i.e. the number of non-zero elements is about 0,1% or less) and the large scale methods are very efficient in time (Zhang 1995). The transformed problem consists of 1620 variables. LP problem has been solved using MATLAB®'s Large Scale Algorithm with additional heuristic (method I) and binary integer algorithm (method II – in this case variable values are limited to 0's and 1's)).

In table 3 value of *Sat(xopt)* for different number of planning steps is presented. The value 1 is for 8 planning steps, so 8 steps are needed to solve this problem. In table 4 value of *U(xopt)* is presented. The value 1 is achieved in 8th iteration of heuristic, so the proposed heuristic is 8 times repeated. In table 5 the comparison of computational efficiency is presented.

Table 3. Value of *Sat(xopt)* for different number of planning steps

Planning step	1	2	3	4	5	6	7	8
Sat(xopt)	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1

Table 4. Value of *U(xopt)*

Iter	1	2	3	4	5	6	7	8
U	0.15	0.19	0.25	0.33	0.44	0.61	0.80	1.00

Table 5. Time of solving planning problem

-	LP + heuristic	Binary integer
Time	about 10 seconds	about 10 minutes

It should be mentioned that such heuristic can not be applied as a general method for finding solution of a linear binary integer problem, because it refers to STRIPS planning problem. It implies that one part of linear program variables correspond to conditions and second part to actions. In proposed heuristic changes (roundings) are made only for action variables (see step ii of the heuristic), but but because of constraints these changes influence condition variable values.

It is also easy to see that classical method based on approximation of binary integer linear program by rounding solution of LP relaxation can lead to infeasible results. This property is shown in table 6, where part of solutions returned by different methods are presented: variable values for *Move-to-table*(*x,y*) action in first plan step for problem in figure 1. LP relaxation returns values 0.5 (third column) – it can be interpreted as an alternative: move block A from B or F from G. LP relaxation with heuristic (fourth column) returns 1 and 0, binary-integer linear program (fifth column) returns 0 and 1, and rounding method (last column) returns 1 and 1 – constraint that one action can be performed in one step is broken (constraint (5)).

Table 6. Solution for different methods

step	action	LP	LP + heur.	Bin.int .LP	Rounding
1	Move-to-table(A,B)	0,5	1	0	1
1	Move-to-table(F,G)	0,5	0	1	1

5.1 Case of incomplete information about initial state

The popular approach of modelling uncertainty environment in STRIPS planning is to treat the initial situation as a set of possible initial states. The problem is that in the general case the number of possible initial states can growth exponentially with the number of uncertain predicates. Such problem is called the problem of planning in the presence of incompleteness (Weld et al. 1998) and is usually much more difficult to solve than 'complete' problem: it belongs to the next level in complexity hierarchy than corresponding problem with complete information (Baralet al. 2000).

It is also impossible to transform polynomially such problem to linear program, because the number of variables exponentially depends of the size of planning problem. To reduce this number of variables of linear program one could apply Kleene's logic system in order to formulate planning system with partly undecided initial state. In this system the valuation space is the three-point set: {0, 1/2, 1} under the usual arithmetic ordering and, intuitively, T(a) = 1, T(a) = 0, and T(a) = 1/2 mean that ``a'' is true, false, and undecided, respectively (Dougherty and Giardina 1988).

Recall the case of 4 blocks Block World problem and assume that the initial state now consists of two possible initial situations (fig.3):

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

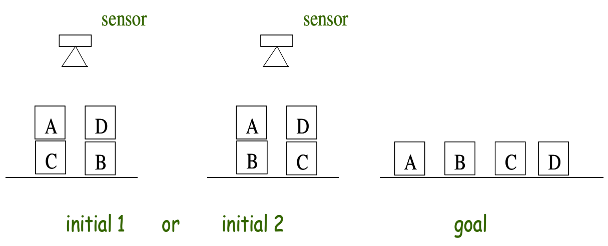

Fig.3. Example 2: The problem with incomplete information in the initial state

Example 2. Using Kleene's Three Valued Logic System the truth values of variables on(A,B)(0), on(D,C)(0), on(A,C)(0), on(D,B)(0) are set to 0.5 what represents the uncertainty whether block A is on C or B and block D is on C or B.

Now the solution corresponds to truth values of actions:

MoveToTable(A,B)(0) = 0.5
MoveToTable(A,C)(0) = 0.5
MoveToTable(D,B)(1) = 0.5
MoveToTable(D,C)(1) = 0.5

that solve planning problem also in 2 steps (see Table 7 – numbers correspond to condition and action variables from table 1). The interpretation of this solution is as follow: first move block A to the table (from B or C) then move block D (from B or C). Please note that in this case the *U(G)* = 1.

Table 7. Solution of Example 2.

	conditions i = 0		*actions i = 0*		*conditions i = 1*		*actions i = 1*		*conditions i = 2*
1.	0.5000	17.	0.5000	29.	0.0000	45.	0.0000	57.	0.0000
2.	0.5000	18.	0.5000	30.	-0.0000	46.	-0.0000	58.	-0.0000
3.	-0.0000	19.	-0.0000	31.	0.0000	47.	0.0000	59.	0.0000
4.	0.0000	20.	0.0000	32.	0.0000	48.	0.0000	60.	-0.0000
5.	-0.0000	21.	-0.0000	33.	-0.0000	49.	-0.0000	61.	-0.0000
6.	0.0000	22.	-0.0000	34.	0.0000	50.	-0.0000	62.	0.0000
7.	-0.0000	23.	-0.0000	35.	0.0000	51.	0.0000	63.	-0.0000
8.	-0.0000	24.	-0.0000	36.	0.0000	52.	0.0000	64.	-0.0000
9.	-0.0000	25.	0.0000	37.	0.0000	53.	-0.0000	65.	0.0000
10.	-0.0000	26.	-0.0000	38.	-0.0000	54.	0.0000	66.	-0.0000
11.	0.5000	27.	0.0000	39.	0.5000	55.	0.5000	67.	0.0000
12.	0.5000	28.	0.0000	40.	0.5000	56.	0.5000	68.	0.0000
13.	1.0000			41.	1.0000			69.	1.0000
14.	-0.0000			42.	0.5000			70.	1.0000
15.	-0.0000			43.	0.5000			71.	1.0000
16.	1.0000			44.	1.0000			72.	1.0000

5.2 Case of uncertain outcomes of actions

The uncertain outcome of an action is modelled in linear program by introducing weight for each action:

$$w_i \in [0,1]$$
$$i = 1,2...k$$

where k is the number of actions O.

Under this notation the value of $w_i = 0.5$ means that the effect of the action i is unknown. It leads to changes in equalities (11) and (12) that define changes in the state after applying an action:

clear(A)(i+1) = clear(A)(i) + w_1 *MoveToTable(B,A)(i)+*
w_2 *MoveToTable(C,A)(i)+* w_3 *MoveToTable(D,A)(i)*

and similar for blocks: *B, C, D*, and:

on(A,B)(i+1) = on(A,B)(i) - w_4 *MoveToTable(A,B)(i).*

Please note that inequalities are not changed (please remember that they describe if an action can be applied to the current state of the problem): only the outcomes of actions are uncertain, not the act of choosing of an action.

In this case *U(G)* is lower than 1 if in the plan actions with w_i lower than 1 are used.

6 Conclusion

Translation to Linear Programming is a heuristic that allows to reduce computational efficiency of searching for the solution. The cost of this approach is that algorithm result can be a non-interpretable solution for some initial states (what is followed by assumption $P \neq NP$). This is because the solution of LP problem is the vector of possibly-continuous values from 0 to 1.

The fastest way to solve transformed planning problem is to use classic LP representation (method I). But in this case the solution can be directly noninterpretable as a plan. This is because the solution of LP problem is the vector of possibly-continuous values from 0 to 1. So additional heuristic to receive the plan from the vector is needed. Binary integer algorithm (method II) returns the plan as a result, but the method is limited to only small sizes of planning problems. This is because the algorithm is exponentially efficient in time (this property is not a conclusion from the table but it is the property of the algorithm).

Scoring functions allow to estimate if the LP solution is a ‘perfect’ solution (i.e. if it can be directly treated as the planning problem solution). The LP solution can be ‘imperfect’ for 2 reasons: 1) it is not binary vector (i.e. does not correspond to true/false conditions and actions), 2) the variable values that correspond to state in last step of planning does not correspond to conditions for the planning problem goal situation (i.e. the LP solution does not lead to goal of planning). Proposed scoring functions have common property: they are equal to1 if the plan is ‘perfect’ due to scored property, and are lower than 1 in the opposite case.

Acknowledgement. This work has been funded from the Polish Ministry of Science and Higher Education funds in years 2008-2010 and is an extended version of the paper presented on the 9th WSEAS Int. Conf. on Simulation, Modelling and Optimization, Budapest 3-5 September 2009.

References

1. Baral Ch., V. Kreinovich, R.Trejo. 2000. Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence*, 122: 241-267
2. Bish, E.K., Leong, T.Y., Li, C.L., Ng, J.W.C., Simchi-Levi, D. 2001. Analysis of a new vehicle scheduling and location problem, *Naval Research Logistics* 48, 363-385
3. Bylander, T. 1994. The computational complexity of propositional STRIPS planning. Artificial Intelligence, 69:165-204.

4. Bylander T. 1997. A Linear Programming Heuristic for Optimal Planning. Int. Conf. American Association for Artificial Intelligence, 1997, www.aaai.org.
5. Galuszka A., A. Swierniak. 2004. Game Theoretic Approach to Multi-Robot Planning. WSEAS Transactions on Computers, ISSN 1109-2750, Issue 3, Volume 3, July 2004: 537-542
6. Galuszka A. 2007. Linear and Integer Programming Large Scale Heuristic for STRIPS Planning. The 2007 European Simulation and Modelling Conference, ESM, October 22-24, 2007, Malta, 321-325.
7. Howe A.E. i E. Dahlman. 2002. A Critical Assesment of Benchmark Comparison in Planning. Journal of Artificial Intelligence Research, 17: 1-33.
8. Chen Kuentai, Hung-Chun Chen, Z.H. Che. 2008. Simulation of Production and Transportation Planning With Uncertainty And Risk. WSEAS Transactions on Computers, Issue 10, Volume 7: 1535-1544.
9. Imai, A., Nishimura, E., Papadimitriou, S. 2001. The dynamic berth allocation problem for a container port, *Transportation Research B* 35, 401-417.
10. Karadimas Nikolaos V. , Nikolaos Doukas, Maria Kolokathi, Gerasimoula Defteraiou, 2008. Routing Optimization Heuristics Algorithms for Urban Solid Waste Transportation Management, WSEAS Transactions on Computers, Issue 12, Volume 7: 2022-2031.
11. Nareyek A., E.C. Freuder , R.Fourer , E. Giunchiglia , R.P. Goldman , H. Kautz , J. Rintanen , A. Tate. 2005. Constraints and AI Planning, IEEE Intelligent Systems, v.20 n.2, p.62-72, March 2005
12. Nebel B. i J. Koehler. 1995. Plan reuse versus plan generation: a theoretical and empirical results. Artificial Intelligence, 76: 427-454.
13. Nilson N. J. 1980. Principles of Artificial Intelligence. Toga Publishing Company, Palo Alto, California.
14. Popovic, D. i V.P. Bhatkar. 1994. Methods And Tools For Applied Artificial Intelligence. Marcel Dekker, Inc., New York, NY.
15. Rintanen, J. 1999. Constructing Conditional Plans by a Theorem-Prover. Journal of Artificial Intelligence Research, 10: 323-352.
16. 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.
17. Slaney J. i S. Thiebaux. 2001. Block World revisited. Artificial Intelligence, 125: 119-153.
18. Vossen T., M. Ball, A. Lotem, D.Nau. 2000. Applying Integer Programming to AI Planning. Knowledge Engineering Review, Volume 15, Issue 1, 85-100.
19. Weld, D.S. 1999. Recent Advantages in AI Planning. AI Magazine.
20. Wilson, I.D., Roach, P.A. 2000. Container stowage planning: a methodology for generating computerised solutions, *Journal of the Operational Research Society* 51, 1248-1255
21. Yen, J., R. Langari, L.A. Zadeh. 1995. Industrial Applications of Fuzzy Logic and Intelligent Systems. IEEE Press. New York.