Archives of Control Sciences
Volume 21(LVII), 2011
No. 3, *pages 243–267*

On transformation of STRIPS planning to linear programming

ADAM GALUSZKA

STRIPS language is a convenient representation for artificial intelligence planning problems. Planning is a task of coming up with a sequence of actions that will achieve a goal. In this work a heuristic of polynomial transformation of STRIPS planning problem to linear programming problem (LP) is presented. This is done because planning problems are hard computational problems (PSPACE- complete in general case) and LP problems are known to be computational easy. Representation of STRIPS planning as a set of equalities and inequalities based on the transformation is also proposed. The exemplary simulation shows the computational efficiency of solving planning problem with proposed transformation.

Key words: planning, problem solving, block world, uncertainty, linear programming, computational complexity

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 (Russel and Norvig 2003, Krogt 2008). Planning is a problem of finding 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. The situation becomes even more complicated when a planner does not have a complete set of information about an environment for which the plan should be created (e.g. Madronero et al. 2010). This is called uncertainty and is essential for exact description of a real environment. There exist large number of different approaches and heuristics that try to deal with planning with uncertainty depending on its kind (e.g. Bhattacharya and Vasant 2007). One can find examples of planning applications in manufacturing, production planning (e.g. Peidro and Vasant 2011), logistics and agentics (eg. Elamvazuthi et al. 2010).

The Authors are with Institute of Automatic Control, Silesian University of Technology, Gliwice, Poland. E-mail: adam.galuszka@polsl.pl

This work has been funded from the Silesian University of Technology grant No RGH-2/RAu0/2011 in years 2011-2012.

Received 10.08.2011. Revised 15.10.2011.

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, Bartak 2008). 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). 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). One should mention STRIPS system is no longer used in its original form. The advanced version of STRIPS introduced in 1987 (Pednault 1994) is called Action Description Language (ADL) and other extensions of planning languages are standardized by Planning Domain Definition Language (PDDL) formalism (Ghallab M. et al. 1994).

In this paper, 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, Galuszka and Swierniak 2010) 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).

1.1. Motivation

Finding a plan is generally a hard computational problem and needs a lot of resources. This hardness is especially characteristic for domain-independent algorithms (Ambite and Knoblock 2001) and it corresponds to difficulties with constructing general solvers. However, it should be noted that even for methods specific for certain domains (e.g. for block world), planning problems usually remain difficult (Bylander 1994). The complexity of planning problems strongly depends on the complexity of the actions defined for the assumed domain (also Bylander 1994). Moreover in real world applications knowledge about environment is incomplete, uncertain and approximate. It implies that planning in the presence of uncertainty is more complex than classical planning.

In general, planning with complete information is *PSPACE*-complete problem. In Block World the problem of optimal planning is *NP*-complete (also Bylander 1997). Planning in the presence of incompleteness is much more complicated (Blythe J. 1999) and belongs to the next level in the hierarchy of completeness. Precisely speaking, if the incompleteness is modeled by a set of possible initial states, then planning problem is NP^{NP}-complete or Σ_2P-complete (Baral et al. 2000).

High level of computational complexity of planning is the reason that different heuristic techniques are very popular in planning solver implementations (e.g. Rosa et al. 2011). One of heuristics is a transformation of planning to Linear Programming prob-

lem. 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. Another approach proposed by Bylander (1997) is to introduce additional linear constraints to LP problem. It allows to solve optimally some class of difficult planning problems using classical LP polynomial algorithms. The cost was not always the LP solution can be interpreted as a plan (what is followed by assumption $P \neq NP$). Also, the size of LP problem increased (polynomially) very fast with the number of planning problem variables.

1.2. Contribution

In this paper a heuristic of polynomial transformation of STRIPS planning problem to linear programming problem (LP) is used (Bylander 1997). This is done because LP problems are known to be computational easy (Chaczijan 1979). The heuristic is modified by cutting some linear constraints and introducing an additional heuristic, that returns the plan. Also, it is proposed how to model uncertain information in planning problem by linear constraints of transformed problem.

The following problems are presented and analyzed:

- polynomial transformation of Block World to Linear Programming,
- handling uncertain information by the transformation,
- a possibility of increasing efficiency by transformation of Linear Program to a set of equalities and inequalities.

1.3. Organization of the paper

The paper is organized as follow: In section 2 the STRIPS language and block world environment is defined. In section 3 the transformation of STRIPS planning to a Linear Programming and an additional heuristic is described. In section 4 a possibility of handling uncertainty is shown. In section 5 transformation of the planning problem to a set of equalities and inequalities is presented. Exemplary simulations are shown in section 6. All is concluded in section 7.

2. STRIPS system and Block World

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 $\langle o_1, o_2, \ldots, o_n \rangle$ 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*:

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

and

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

3. 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:

$$\begin{aligned} &\textit{move-to-table}(x,y): \\ &\qquad \textit{preconditions}: on(x,y), clear(x) \\ &\qquad del: on(x,y) \\ &\qquad add: clear(y). \end{aligned} \tag{3}$$

The goal is reached if the following conditions are true:

$$G = clear(A), clear(B), clear(C), clear(D). \tag{4}$$

There are needed 16 conditions in set C to describe state of the problem:

$$\begin{aligned} C = \{ \quad & on(A,B), on(A,C), on(A,D), on(B,A), on(B,C), on(B,D), on(C,A), \\ & on(C,B), on(C,D), on(D,A), on(D,B), on(D,C), clear(A), clear(B), \\ & clear(C), clear(D) \quad \}, \end{aligned} \tag{5}$$

and 12 actions in set O that correspond to 12 conditions $on(x,y)$. Under assumption, that the number of planning steps $l = 2$, the number of states is $l+1 = 3$. So there are needed $3 \cdot 16$ variables for conditions (state indices are $i = 0,1,2$), and $2 \cdot 12$ conditions for actions (indices for actions are $i = 0,1$) in LP problem. The number of all variables x of LP is 72, and the variables are presented in Tab. 1.

It should be noted that conditions and actions parameters are extended by indices of planning step. The goal function to be maximized in LP $F(G)$ consists of conditions that are true in goal state:

$$F(G) = (clear(A,2) + clear(B,2) + clear(C,2) + clear(D,2)). \tag{6}$$

If the goal is satisfied then $F(G) = 4$. Linear constraints followed by STRIPS action definition (3) are:

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

$$\Sigma \ \textit{move-to-table}(x,i) = 1 \tag{7}$$

- action cannot be applied unless its preconditions are true:

$$on(x,y,i) \geqslant \textit{move-to-table}(x,y,i) \tag{8}$$

for all actions in each planning step,

$$\begin{aligned} clear(A,i) \quad &\geqslant \quad \textit{move-to-table}(A,B,i) \\ &+ \quad \textit{move-to-table}(A,C,i) \ + \ \textit{move-to-table}(A,D,i), \end{aligned} \tag{9}$$

Table 1. Variables of LP problem in each step

conditions $i = 0$	actions $i = 0$	conditions $i = 1$
x1: on(A,B,0)	x17: move-to-table(A,B,0)	x29: on(A,B,1)
x2: on(A,C,0)	x18: move-to-table(A,C,0)	x30: on(A,C,1)
x3: on(A,D,0)	x19: move-to-table(A,D,0)	x31: on(A,D,1)
x4: on(B,A,0)	x20: move-to-table(B,A,0)	x32: on(B,A,1)
x5: on(B,C,0)	x21: move-to-table(B,C,0)	x33: on(B,C,1)
x6: on(B,D,0)	x22: move-to-table(B,D,0)	x34: on(B,D,1)
x7: on(C,A,0)	x23: move-to-table(C,A,0)	x35: on(C,A,1)
x8: on(C,B,0)	x24: move-to-table(C,B,0)	x36: on(C,B,1)
x9: on(C,D,0)	x25: move-to-table(C,D,0)	x37: on(C,D,1)
x10: on(D,A,0)	x26: move-to-table(D,A,0)	x38: on(D,A,1)
x11: on(D,B,0)	x27: move-to-table(D,B,0)	x39: on(D,B,1)
x12: on(D,C,0)	x28: move-to-table(D,C,0)	x40: on(D,C,1)
x13: clear(A,0)		x41: clear(A,1)
x14: clear(B,0)		x42: clear(B,1)
x15: clear(C,0)		x43: clear(C,1)
x16: clear(D,0)		x44: clear(D,1)

actions $i = 1$	conditions $i = 2$
x45: move-to-table(A,B,1)	x57: on(A,B,2)
x46: move-to-table(A,C,1)	x58: on(A,C,2)
x47: move-to-table(A,D,1)	x59: on(A,D,2)
x48: move-to-table(B,A,1)	x60: on(B,A,2)
x49: move-to-table(B,C,1)	x61: on(B,C,2)
x50: move-to-table(B,D,1)	x62: on(B,D,2)
x51: move-to-table(C,A,1)	x63: on(C,A,2)
x52: move-to-table(C,B,1)	x64: on(C,B,2)
x53: move-to-table(C,D,1)	x65: on(C,D,2)
x54: move-to-table(D,A,1)	x66: on(D,A,2)
x55: move-to-table(D,B,1)	x67: on(D,B,2)
x56: move-to-table(D,C,1)	x68: on(D,C,2)
	x69: clear(A,2)
	x70: clear(B,2)
	x71: clear(C,2)
	x72: clear(D,2)

$$\begin{aligned} clear(B,i) &\geqslant move\text{-}to\text{-}table(B,A,i) \\ &+ move\text{-}to\text{-}table(B,C,i) + move\text{-}to\text{-}table(B,D,i), \end{aligned} \tag{10}$$

$$\begin{aligned} clear(C,i) &\geqslant move\text{-}to\text{-}table(C,A,i) \\ &+ move\text{-}to\text{-}table(C,B,i) + move\text{-}to\text{-}table(C,D,i), \end{aligned} \tag{11}$$

$$\begin{aligned} clear(D,i) &\geqslant move\text{-}to\text{-}table(D,A,i) \\ &+ move\text{-}to\text{-}table(D,B,i) + move\text{-}to\text{-}table(D,C,i). \end{aligned} \tag{12}$$

for all planning steps.

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

$$\begin{aligned} clear(A,i+1) &= clear(A,i) + move\text{-}to\text{-}table(B,A,i) \\ &+ move\text{-}to\text{-}table(C,A,i) + move\text{-}to\text{-}table(D,A,i) \end{aligned} \tag{13}$$

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

$$on(A,B,i+1) = on(A,B,i) - move\text{-}to\text{-}table(A,B,i). \tag{14}$$

The initial state of the problem is also mapped into equality constraints:

$$on(x,y,0) = \begin{cases} 1, & \text{if} \quad on(x,y,0) \in G; \\ 0, & \text{in opposite case,} \end{cases} \tag{15}$$

$$clear(x,0) = \begin{cases} 1, & \text{if} \quad clear(x,0) \in G; \\ 0, & \text{in opposite case.} \end{cases} \tag{16}$$

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:

$$p = (l+1)(2n2+n), \tag{17}$$

- the number of inequality constraints:

$$l(n2+n+1), \tag{18}$$

- the number of equality constraints:

$$l(n2+n), \tag{19}$$

so the transformation is polynomial.

If the LP problem is defined as:

$$\begin{aligned} \max_x \leftarrow f^T x \qquad & AX \leqslant b \\ & A_{eq}x = b_{eq} \\ & 0 \leqslant x \leqslant 1 \end{aligned} \tag{20}$$

then f defines the goal state of the planning problem, and the initial state is enclosed in A_{eq} and b_{eq}. The vector x consists of all LP variables presented in Tab. 1, and the solution of LP problem is denoted by x_{opt}. Matrix A_{eq} and vector b_{eq} correspond to equality constraints (13-16), matrix A and vector b correspond to inequality constraints (7-12). Matrices A and A_{eq} which correspond to the planning problem are sparse and their nonzero elements are illustrated in the Fig. 1.

The parameter vector f of the goal function consists of '1' for variables that correspond to true conditions in goal state, so for goal state (4) and corresponding goal function (6):

$$f = (\text{zeros}(1,68),\ 1\ 1\ 1\ 1). \tag{21}$$

3.1. Example 1

To illustrate the transformation Example 1 is shown. The problem (Fig. 2) is to decompose the initial state defined by the set: *on(A,B), on(D,C), clear(A), clear(D).*

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 (Tab. 2).

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 $P \neq NP$). 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 has been introduced in section 3.2, and the additional heuristic has been proposed in section 3.3.

3.2. 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:

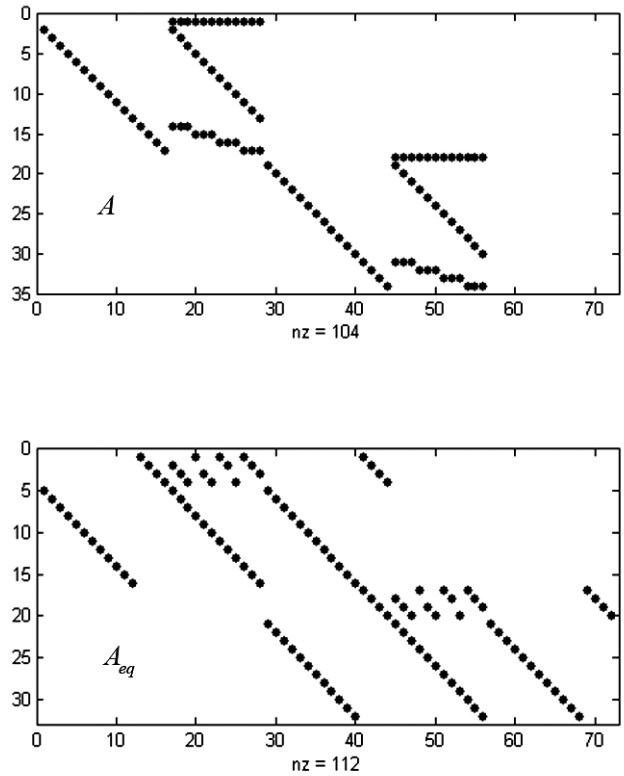

Figure 1. Matrices for linear program of Example 1.

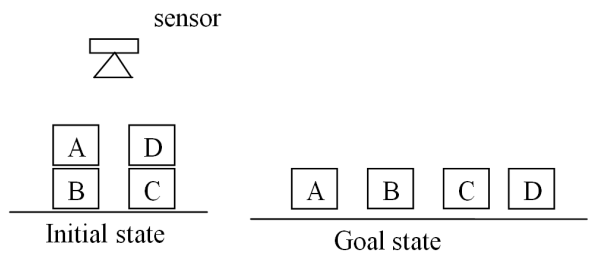

Figure 2. The initial and goal states for Example 1.

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).

Table 2. Solution of Example 1.

conditions $i=0$	actions $i=0$	conditions $i=1$	actions $i=1$	conditions $i=2$
x1: 1.00	x17: -0.00	x29: 1.00	x45: 1.00	x57: 0.00
x2: 0.00	x18: -0.00	x30: 0.00	x46: 0.00	x58: -0.00
x3: 0.00	x19: 0.00	x31: 0.00	x47: 0.00	x59: 0.00
x4: 0.00	x20: 0.00	x32: 0.00	x48: 0.00	x60: -0.00
x5: -0.00	x21: -0.00	x33: -0.00	x49: -0.00	x61: -0.00
x6: -0.00	x22: 0.00	x34: 0.00	x50: 0.00	x62: 0.00
x7: 0.00	x23: 0.00	x35: -0.00	x51: -0.00	x63: 0.00
x8: -0.00	x24: 0.00	x36: 0.00	x52: -0.00	x64: -0.00
x9: 0.00	x25: -0.00	x37: 0.00	x53: 0.00	x65: 0.00
x10: 0.00	x26: 0.00	x38: -0.00	x54: -0.00	x66: 0.00
x11: -0.00	x27: 0.00	x39: -0.00	x55: -0.00	x67: 0.00
x12: 1.00	x28: 1.00	x40: 0.00	x56: -0.00	x68: 0.00
x13: 1.00		x41: 1.00		x69: 1.00
x14: -0.00		x42: 0.00		x70: 1.00
x15: 0.00		x43: 1.00		x71: 1.00
x16: 1.00		x44: 1.00		x72: 1.00

Proposed scoring functions have common property: they are equal to 1 if the plan is 'perfect' due to scored property, and are lower than 1 in the opposite case (Galuszka 2009). The first function, $U(xopt)$, called $planutility$, estimates how close is the LP solution to binary solution:

$$U(x_{opt}) = \frac{l(x_{opt})}{k(x_{opt})} \tag{22}$$

where: l – number of planning steps, k – number of non-zeros variables of LP solution that correspond to actions, x_{opt} – LP problem optimal solution.

The second function, $Sat(x_{opt})$, 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:

$$Sat(x_{opt}) = \frac{f^T x_{opt}}{m} \tag{23}$$

where: $f^T x_{opt}$ – optimal 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. $Sat(x_{opt})$ has the property that returns 1 if the plan satisfies all predicates in the goal definition, and 0 in the opposite case. Value of $Sat(x_{opt})$ between 0 and 1 indicates that there are some variables that correspond to goal state not equal to 1, what follows that goal state is not reached.

3.3. Additional heuristic

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

a) Solve the Linear Programming problem that corresponds to planning problem;

b) From the group of variables that correspond to operators in the 1st step choose the one with the biggest value of truth degree;

c) Assume the value of this variable equal to 1 (it means that this operator is chosen as the best proposition) and add this equality as an additional constraint to the LP problem;

d) Solve this more restricted problem;

e) 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 et al. 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.

3.4. Remarks to the heuristic

The result of the heuristic is a binary solution of the linear program that represents 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 first step of the heuristic an additional equality constraint that correspond to 1 action in first planning step is added (step c) of the heuristic), so from constraints (5) variables for all actions in this step are set to 0, that corresponds to one action.

4. Modeling uncertainty in Linear Program

In many real world applications an initial state of the problem given as a set of predicates is, realistically speaking, an overidealization since model and measurement inaccuracies, disturbances and imperfect processing procedures result in the uncertainty in the problem variables.

Modern planners have developed techniques that allow planner to find plans in situation where there can be many sources of uncertainty and reasons efficiently about these plans (but they are still exponentially efficient in time) (Blythe 1999). They can handle uncertainty from three types of cause

- incomplete information about initial state,
- uncertain outcomes of actions,
- non-deterministic exogenous events.

The third cause of uncertainty is very important because external events can very often play an important role in real domains. Theoretically, it could be done by modeling effects of exogenous events by effects of actions in the domain. This is, however, very inefficient way, since their effects are duplicated in every action in the domain. In such a way each action might have to model many events.

Below it is shown how to implement first two kinds of uncertainty in linear program.

4.1. Case of incomplete information about initial state

Recall the case of 4 blocks Block World problem and assume that the initial state now can be modeled basing on sensor information (Galuszka 2006). It implies that positions of blocks B and C are indistinguishable and cannot be precisely described in the initial state definition. It leads to two possible initial situations (Fig. 3):

$$(on(A,B),\ on(D,C),\ clear(A),\ clear(D))$$

or

$$(on(A,C),\ on(D,B),\ clear(A),\ clear(D)).$$

The popular approach of modeling 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 (Baral et 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

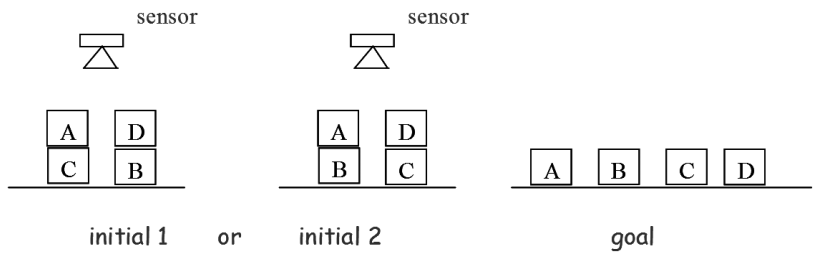

Figure 3. Example 2: The problem with incomplete information in the initial state.

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).

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:

$$\begin{aligned} &\textit{move-to-table}(A,B,0) = 0.5,\ \ \textit{move-to-table}(A,C,0) = 0.5, \\ &\textit{move-to-table}(D,B,1) = 0.5,\ \ \textit{move-to-table}(D,C,1) = 0.5 \end{aligned} \tag{24}$$

that solve planning problem also in 2 steps (see Tab. 3 – numbers correspond to condition and action variables from Tab. 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). Note that in this case $U(x_{opt}) = 1$.

4.2. Case of uncertain outcomes of actions

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

$$w_i \in [0,1] \tag{25}$$

where $i = 1,2,\ldots,k$, and 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 (1) and (2) that define changes in the state after applying an action:

Table 3. Solution of Example 2.

conditions $i=0$	actions $i=0$	conditions $i=1$	actions $i=1$	conditions $i=2$
x1: 0.50	x17: 0.50	x29: 0.00	x45: 0.00	x57: 0.00
x2: 0.50	x18: 0.50	x30: 0.00	x46: 0.00	x58: 0.00
x3: 0.00	x19: 0.00	x31: 0.00	x47: 0.00	x59: 0.00
x4: 0.00	x20: 0.00	x32: 0.00	x48: 0.00	x60: 0.00
x5: 0.00	x21: 0.00	x33: 0.00	x49: 0.00	x61: 0.00
x6: 0.00	x22: 0.00	x34: 0.00	x50: 0.00	x62: 0.00
x7: 0.00	x23: 0.00	x35: 0.00	x51: 0.00	x63: 0.00
x8: 0.00	x24: 0.00	x36: 0.00	x52: 0.00	x64: 0.00
x9: 0.00	x25: 0.00	x37: 0.00	x53: 0.00	x65: 0.00
x10: 0.00	x26: 0.00	x38: 0.00	x54: 0.00	x66: 0.00
x11: 0.50	x27: 0.00	x39: 0.50	x55: 0.50	x67: 0.00
x12: 0.50	x28: 0.00	x40: 0.50	x56: 0.50	x68: 0.00
x13: 1.00		x41: 1.00		x69: 1.00
x14: 0.00		x42: 0.00		x70: 1.00
x15: 0.00		x43: 1.00		x71: 1.00
x16: 1.00		x44: 1.00		x72: 1.00

$$\begin{aligned} clear(A,i+1) &= clear(A,i) + w_1 move\text{-}to\text{-}table(B,A,i) \\ &+ w_2 move\text{-}to\text{-}table(C,A,i) + w_3 move\text{-}to\text{-}table(D,A,i) \end{aligned} \tag{26}$$

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

$$on(A,B,i+1) = on(A,B,i) - w_4 move\text{-}to\text{-}table(A,B,i). \tag{27}$$

Note that inequalities are not changed (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 $U(x_{opt}) < 1$ if in the plan actions with $w_i < 1$ are used.

4.3. Handling sensory actions: Example 3

One of the way of dealing with uncertainty in planning is to extend the set of operators with sensory actions (Weld et al. 1998). After executing this sensory action the

planner knows whether sensed predicate is true or not in the current problem state. It allows to resolve uncertainty during execution of the plan.

As Example 3 let us recall Example 2 of planning in Block World environment with 4 blocks (called A, B, C, D – see Fig. 3). The goal is to decompose the initial state. Now 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:

$$\begin{aligned} &move\text{-}to\text{-}table(x): \\ &preconditions: on(x), clear(x) \\ &del: on(x) \\ &add: clear(y) \end{aligned} \tag{28}$$

The sensory action

$$\begin{aligned} &sensing-block(x): \\ &preconditions: (noprecond.) \\ &add: clear(x) \end{aligned} \tag{29}$$

is an action that allows to sense what block is under top block after moving top block to the table.

The transformation of problem with sensory action to linear program is now different. Assume 4 steps of planning (states indexes are: 0, 1, 2, 3, 4). Then the objective function is:

$$Max(clear(A,4)+clear(B,4)+clear(C,4)+clear(D,4)). \tag{30}$$

If the goal is reached then the objective function is equal to 4 (4 conditions are true in the goal state). There are also some changes in constraints of Linear Programming problem. They are because a sensory action is performed after a block is moved to the table:

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

$$\Sigma\ move\text{-}to\text{-}table(x,i) = 1 \tag{31}$$

  for even $i = 0,2,4$, and:

$$\Sigma\ sensing-block(x,i) = 1 \tag{32}$$

  for odd $i = 1,3$ (after moving to table sensing is performed),

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

$$on(x,i) \geqslant move\text{-}to\text{-}table(x,i) \tag{33}$$

  and

$$clear(x,i) \geqslant move\text{-}to\text{-}table(x,i) \tag{34}$$

  for $i = 0,2,4$.

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

$$clear(x,i+1) = clear(x,i) + sensing - block(x,i) \tag{35}$$

$$on(x,i+1) = on(x,i) - move\text{-}to\text{-}table(x,i). \tag{36}$$

Additional equality constraints are needed to express information about real situation in planning problem, i.e. what blocks are found after performing sensory action. Without these constraints the planner will search for the solution taking into account all possible situation resulted from incomplete information (2^n possible states for n hidden blocks), what is impossible to present by polynomial transformation to LP problem. These constraints take the form:

$$sensing - block(y,i+1) = move\text{-}to\text{-}table(x,i) \tag{37}$$

where block y is the sensed block after moving to table block x.

Finally constraints for variables are needed. The constraints should be mapped between values 0 and 1 what corresponds to truth values of the variables and to binary integer linear programming problem.

Summarizing, the size of LP problem that corresponds to STRIPS problem with sensing action is:

- the number of variables: $3 \cdot n \cdot l + 2 \cdot n$,
- the number of constraints: $l \cdot (4 \cdot n + 1 + n_{hidden})$, where n is the number of blocks, l is the number of planning steps and n_{hidden} is the number of hidden blocks.

For Example 3 ($n = 4$, $l = 4$) there are 56 variables of LP problem. Nonzero elements of matrix of equality constraints ($A \cdot x = b$) and inequality constraints ($A_{eq} \cdot x = b_{eq}$) are presented in Fig. 4.

5. Transformxation of block world planning to a set of equalities and inequalities

Under assumption that values of variables describing the goal state are known, the goal state can be represented by equality constraints. It implies that it is possible to represent the planning problem as a set of equalities and inequalities:

$$\begin{aligned} Cx &= d \\ Ax &\leqslant b \\ 0 &\leqslant x \leqslant 1 \end{aligned} \tag{38}$$

where C is the matrix A_{eq} (20) extended by equalities that defines the goal state variables, and d is the vector b_{eq} (20) extended by values of these variables in the goal state. Matrix

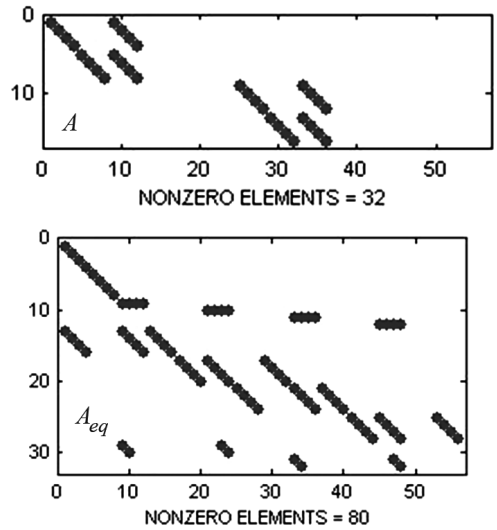

Figure 4. Matrices for linear program of example 3.

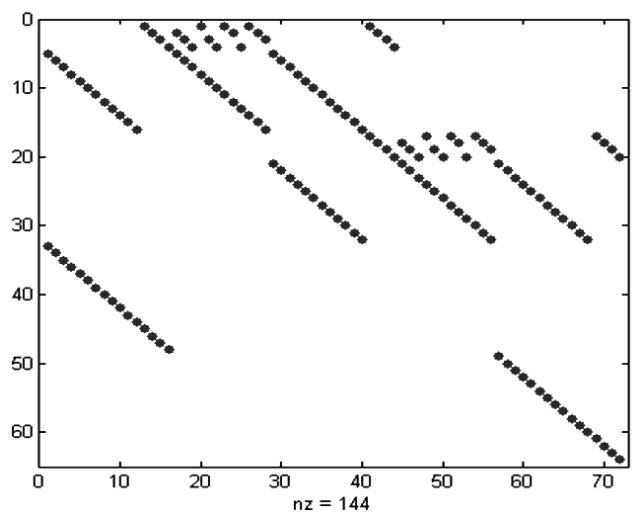

Figure 5. Nonzero elements of matrix C from (30).

A and vector b are equal to those in (20). Nonzero elements of matrix C for the planning problem presented in Fig. 1 is shown in Fig. 5.

The goal state of Example 1 can be express by following equalities (values of variables of 2nd step of planning): $clear(A,2) = 1$, $clear(B,2) = 1$, $clear(C,2) = 1$, $clear(D,2) = 1$. The rest variables are set to 0.

In this figure additional equality constraints needed to express the goal state are placed under the dashed line.

6. Simulations

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 and PC Intel® Core™ i5 CPU, 2,67 GHz, 4GB RAM. The assumed initial state is shown in Fig. 6. It is assumed 8 planning steps.

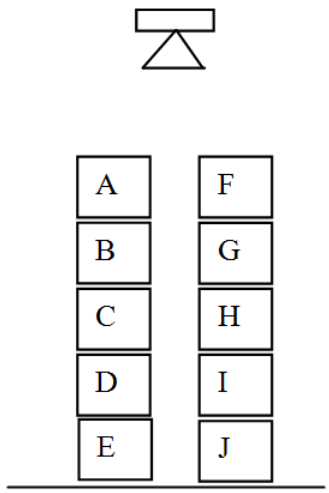

Figure 6. The initial state for an exemplary problem.

In Fig.7. nonzero elements of matrices of 10 block 8 planning step block world problem are presented. 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). In this figure additional equality constraints needed to express the goal state (for representation by a set of equalities and inequalities) are placed under the dashed line (see Fig. 7b).

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 IV - in this case variable values are limited to 0's and 1's)).

For Large Scale Algorithm with additional heuristic the LP program was repeated 8 times with additional constraints (it correspond to the number of planning steps), because vector with many non-zero and non-one values was obtained as a result. In the Fig. 8 non-zero elements of solution vectors are shown: column 1 is for the first solution of LP

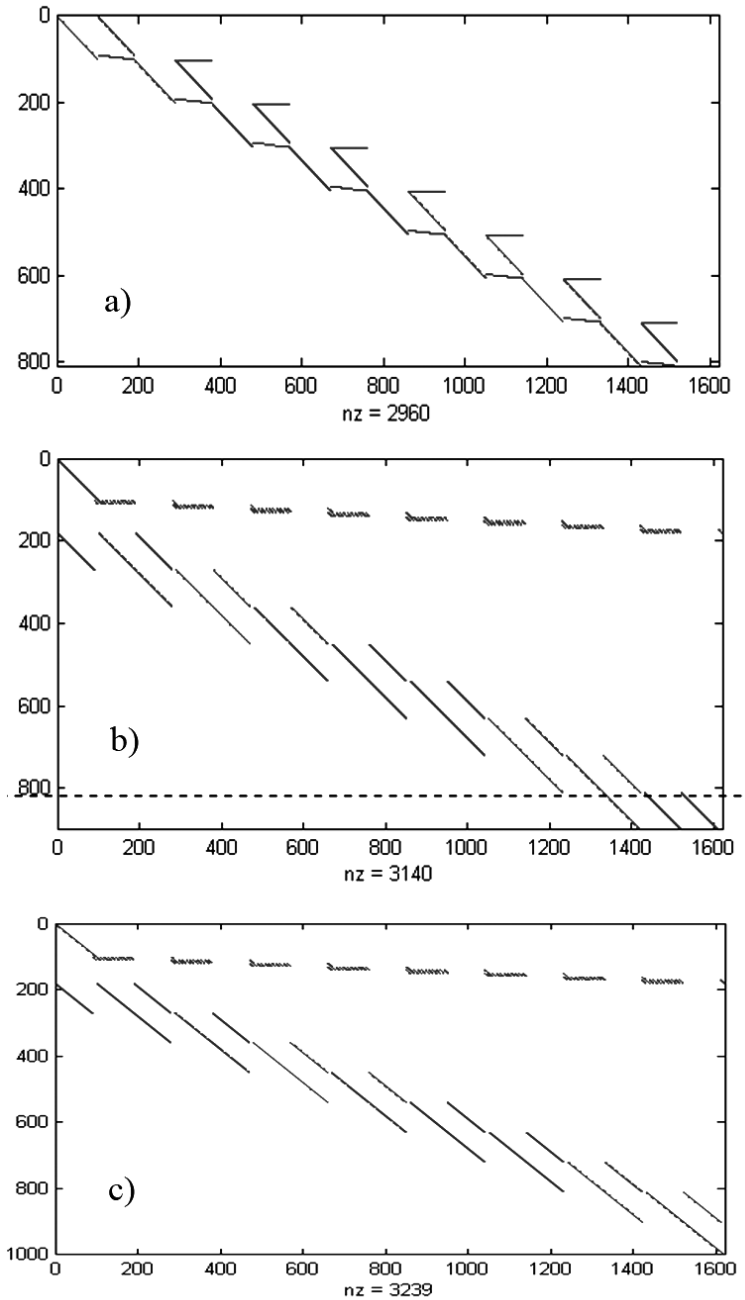

Figure 7. Nonzero elements of matrices of 10 block 8 planning step block world problem: a) matrix A, b) matrix A_{eq}, c) matrix C.

program (many non-zero elements and most of them are also non-one), column 8 is for the last one (less non-zero elements and all of them are 1's).

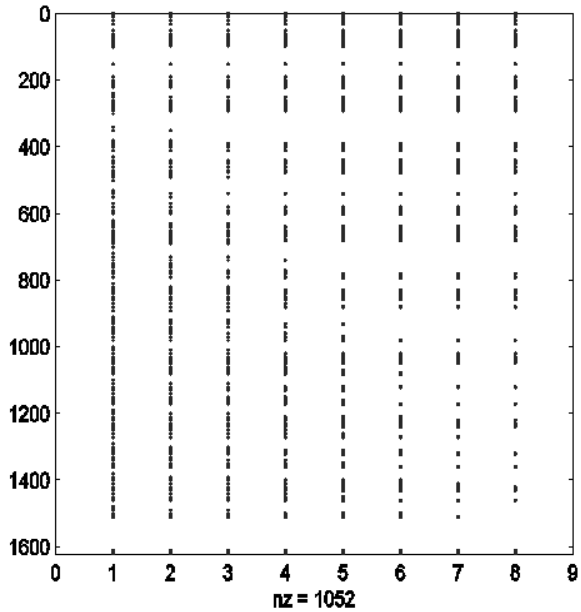

Figure 8. Non-zero elements of solution vectors.

Set of equalities and inequalities has been solved using quadratic programming algorithm (method II):

$$\begin{aligned} \min \leftarrow g &= (Cx-d)^2 \\ Ax &\leqslant b \\ 0 &\leqslant x \leqslant 1 \end{aligned} \tag{39}$$

where the cost function g should be equal to 0 to satisfy equality $Cx = d$.

The modification of the cost function g can also be proposed. The modified cost function includes only 1 variable with known value in the planning problem. Variables with known value are in description of the initial and the goal states of the problem, and one can choose any of these variables. The modified problem (method III) now takes the form:

$$\begin{aligned} \min \leftarrow g^* &= (x^* - d^1)^2 \\ C^* &= d^* \\ Ax &\leqslant b \\ 0 &\leqslant x \leqslant 1 \end{aligned} \tag{40}$$

where g^* is the modified cost function, x^* is variable with known value, d^1 is 0 or 1 (dependent on whether x^* is true or false), C^* is C without column for variable x^*, d^* is d without row with value of x^*.

Simulation result (time of calculating solution for each method) is shown in Tab. 4.

Table 4. Simulation results for 4 different methods (Athlon 2800, 512 MB RAM)

I – linear programming with additional heuristic;

II – quadratic programming;

III – quadratic programming for 1 variable;

IV – binary integer programming

method	**I**	**II**	**III**	**IV**
time[s]	**4**	**499**	**974**	**594**

7. Conclusion

Translation to Linear Programming is a heuristic that allows to reduce computational complexity of searching for the solution. That is because planning in the presence of incompleteness is usual at least NP-complete problem, Linear Programming is polynomial-time complete problem and translation from STRIPS to Linear Programming is also polynomial. The cost of this approach is that algorithm can results in noninterpretable solutions 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. So additional heuristic to receive the plan from the vector is needed.

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 IV) 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).

STRIPS problem as a set of equalities and inequalities is an attractive idea because we receive non-optimization problem as a result. Unfortunately, available numerical methods for solving this are also optimization ones (methods II and III) and very slow.

Finally, results should be read as a comparison of efficiency of different introduced representations of Block World planning, taking into account that the transformation is

polynomial and all methods (except IV) are polynomial time complete. The conclusion that 10 minutes is needed to solve 10 block toy problem is wrong. So the heuristic is polynomial and that is why it reduces computational complexity of finding optimal solution of STRIPS planning problem, but only if this solution of transformed problem can be interpretable as a plan (the is the cost of transformation followed by the assumption $P \neq NP$).

The advantage of proposed heuristic of planning using transformation to LP is that the time of finding problem solution is polynomially limited by the size of the problem. The disadvantage is that transformation can lead to LP problems with large number of variables, what can be difficult to implement even using modern computers. That is why additional research should be performed in order to find efficient ways to reduce the number of LP problem variables.

References

[1] J.L. AMBITE and C.A. KNOBLOCK: Planning by rewriting. *J. of Artificial Intelligence Research*, **15** (2001), 207-261.

[2] M. AVRIEL, M. PENN, N. SHPIRER and S. WITTEBOON: Stowage planning for container ships to reduce the number of shifts. *Annals of Operations Research*, **76** (1998), 55-71.

[3] CH. BACKSTROM: Computational aspects of reordering plans. *J. of Artificial Intelligence Research*, **9** (1998), 99-137.

[4] CH. BARAL, V. KREINOVICH and R.TREJO: Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence*, **122** (2000), 241-267.

[5] R. BARTAK: Constraint satisfaction techniques in planning and scheduling: An introduction. *Archives of Control Sciences*, **18**(2), (2008).

[6] A. BHATTACHARYA and P. VASANT: Soft-sensing of level of satisfaction in TOC product-mix decision heuristic using robust fuzzy-LP. *European J. of Operational Research*, **177**(1), (2007), 55-70.

[7] E.K. BISH, T.Y. LEONG, C.L. LI, J.W.C. NG and D. SIMCHI-LEVI: Analysis of a new vehicle scheduling and location problem. *Naval Research Logistics*, **48** (2001), 363-385.

[8] J. BLYTHE: An overview of planning under uncertainty. *Pre-print from AI Magazine*, **20**(2), (1999), 37-54.

[9] C. BOUTILIER and R.I. BRAFMAN: 2001. Partial-order planning with concurrent interacting actions. *J. of Artificial Intelligence Research*, **14** (2001), 105-136.

[10] T. BYLANDER: The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, **69** (1994), 165-204.

[11] T. BYLANDER: A linear programming heuristic for optimal planning. *In Proc. of AAAI Nat. Conf.*, (1997).

[12] L.G. CHACZIJAN: A polynomial algorithm for linear programming. *Dokl. Akad. Nauk SSSR*, **244** (1979), 1093-1096.

[13] E.R. DOUGHERTY and CH.R. GIARDINA: Mathematical methods for artificial intelligence and autonomous systems. Prentice-Hall International Inc., USA, 1988.

[14] I. ELAMVAZUTHI, P. VASANT and T. GANESAN: Fuzzy linear programming using modified logistic membership function. *Int. Review of Automatic Control (IREACO)*, **3**(4), (2010), 370-377.

[15] A. GALUSZKA and A. SWIERNIAK: Translation STRIPS planning in multi-robot environment to linear programming. *ICAISC 2004 (LNCS 3070)*, (2004), 768-773.

[16] A. GALUSZKA and A. SWIERNIAK: Planning in multi-agent environment using strips representation and non-cooperative equilibrium strategy. *J. of Intelligent and Robotic Systems*, Springer Netherlands, ISSN: 1573-0409, **58**(3), (2010), 239-251.

[17] A. GALUSZKA: Block world planning with uncertainty and sensing actions as integer linear programming problem. A. Cader, L. Rutkowski, R. Tadeusiewicz, J. Zurada (Eds.): Artificial intelligence and soft computing (ISBN 83-60434-09-3), IEEE Comp. Intell. Socielty - Poland, 2006, 389-395.

[18] A. GALUSZKA: Scoring functions of approximation of STRIPS planning by linear programming. In Simulation, Modelling and Optimization. Mathematics and Computers in Science and Engineering. A series of Reference Books and Textbooks (Eds. I. Rudas *et al.*), ISBN 978-960-474-113-7, 2009, 316-321.

[19] M GHALLAB: PDDL - the planning domain definition language. Version 1.2. Technical Report DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

[20] A. IMAI, E. NISHIMURA, and S. PAPADIMITRIOU: The dynamic berth allocation problem for a container port. *Transportation Research B*, **35** (2001), 401-417.

[21] J. KOEHLER, and J. HOFFMANN: On reasonable and forced goal orderings and their Use in an agenda-driven planning algorithm. *J. of Artificial Intelligence Research*, **12** (2000), 339-386.

[22] J. KOEHLER and K. SCHUSTER: Elevator control as a planning problem. *AIPS-2000*, (2000), 331-338.

[23] S. KRAUS, K. SYCARA and A. EVENCHIK: Reaching agreements through argumentation: a logical model and implementation. *Artificial Intelligence*, **104** (1998), 1-69.

[24] R. VAN DER KROGT: Modification strategies for SAT-based plan adaptation. *Archives of Control Sciences*, **18**(2), (2008).

[25] M.D. MADRONERO, D. PEIDRO and P. VASANT: . Vendor selection problem by using an interactive fuzzy multi-objective approach with modified s-curve membership functions. *Computers and Mathematics with Applications*, **60** (2010), 1038-1048.

[26] A. NAREYEK, C. FREUDER, R. FOURER, E. GIUNCHIGLIA, R.P. GOLDMAN, H. KAUTZ, J. RINTANEN and A.TATE: Constraitns and AI planning. *IEEE Intelligent Systems*, (2005), 62-72.

[27] N.J. NILSON: Principles of artificial intelligence. Toga Publishing Company, Palo Alto, CA, 1980.

[28] E.P.D. PEDNAULT: ADL and the state-transition model of action. *J. of Logic and Computation*, **4**(5), (1994), 467-512.

[29] D. PEIDRO and P. VASANT: Transportation planning with modified s-curve membership functions using an interactive fuzzy multi-objective approach. *Applied Soft Computing*, **11** (2011), 2656-2663.

[30] T. ROSA, S. JIMENEZ, R. FUENTETAJA and D. BARRAJO: Scaling up heuristic planning with relational decision trees. *J. of Artificial Intelligence Research*, **40** (2011), 767-813.

[31] S.J. RUSSELL and P. NORVIG: Artificial intelligence: A modern approach. Second Edition. Prentice Hall Series in Artificial Intelligence, 2003.

[32] J. SLANEY and S. THIEBAUX: Block world revisited. *Artificial Intelligence*, **125** (2001), 119-153.

[33] T. SLAVIN: Virtual port of call. *New Scientist*, (1996), 40-43.

[34] D.E. SMITH and D.S. WELD: Conformant graphplan. *Proc. 15th National Conf. on AI*., (1998).

[35] D.S. WELD: Recent advantages in AI planning. *AI Magazine*, (1999).

[36] D.S. WELD, C.R. ANDERSON and D.E. SMITH: Extending graphplan to handle uncertainty and sensing actions. *Proc. 15th National Conf. on AI*, (1998), 897-904.

[37] I.D. WILSON and P.A. ROACH: Container stowage planning: a methodology for generating computerised solutions. *J. of the Operational Research Society*, **51** (2000), 1248-1255.

[38] Y. ZHANG: . Solving large-scale linear programs by interior-point methods under the MATLAB Environment. Technical Report TR96-01, Department of Mathematics and Statistics, University of Maryland, Baltimore County, Baltimore, MD, 1995.