Scoring Functions of Approximation of STRIPS Planning by Linear Programming

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

Abstract: STRIPS planning is a problem of finding a set of actions that transform given initial state to desired goal situation. It is a hard computational problem. In this work an approximation of STRIPS block world planning by linear programming is shown. Additionally two scoring functions have been introduced to estimate quality of the plan. Proposed approach is illustrated by exemplary simulation.

Keywords: STRIPS planning, 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). 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.

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, Galuszka and Swierniak 2005) 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).

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 ≠ 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 description of the initial state 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)).

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):

(1)
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)

(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. Then the value of the objective
function to be maximized F(G) is:

\[ F(G) = (\text{clear(A)}(2) + \text{clear(B)}(2) + \text{clear(C)}(2) + \text{clear(D)}(2)). \]

(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:

\[ \sum \text{move-to-table}(x(i)) = 1 \]

(5)

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

\[ \text{on}(x,y)(i) \geq \text{move-to-table}(x,y)(i) \]

(6)
for all operators in each planning step, e.g. for block A:

\[ \text{clear}(A)(i) \geq \text{move-to-table}(A,B)(i) + \text{move-to-table}(A,C)(i) + \text{move-to-table}(A,D)(i) \]  

(7)

for all planning steps.

Next group of constraints describes changes of the state after applying an operator. These are equality constraints, e.g. for block A:

\[ \text{clear}(A)(i+1) = \text{clear}(A)(i) + \text{move-to-table}(B,A)(i) + \text{move-to-table}(C,A)(i) + \text{move-to-table}(D,A)(i) \]  

(8)

\[ \text{on}(A,B)(i+1) = \text{on}(A,B)(i) - \text{move-to-table}(A,B)(i). \]  

(9)

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.

If the LP problem is defined as:

\[
\begin{align*}
\max & \quad f^T x \\
\text{s.t.} & \quad Ax \leq b \\
& \quad A_{eq} x = b_{eq} \\
& \quad 0 \leq x \leq 1
\end{align*}
\]  

(10)

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 \( x_{opt} \).

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 to 1 if the plan is 'perfect' due to scored property, and are lower than 1 in the opposite case. The first function, \( U(x_{opt}) \), called plan utility, estimates how close is the LP solution to binary solution:

\[ U(x_{opt}) = \frac{l}{k}, \]  

(11)

where: \( l \) – number of the planning steps, \( k \) – number of non-zeros variables of LP solution that correspond to actions, \( x_{opt} \) – LP solution. The second function, \( \text{Sat}(x_{opt}) \), 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:

\[ \text{Sat}(x_{opt}) = \frac{f^T x_{opt}}{m}, \]  

(12)

where: \( f^T x_{opt} \) - 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. \( \text{Sat}(x_{opt}) \) has the property that returns 1 if the plan satisfies all predicates in the goal definition, and 0 in the opposite case. 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 chosen 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 1 value of $Sat(x_{opt})$ 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 2 value of $U(x_{opt})$ is presented. The value 1 is achieved in 8th iteration of heuristic, so the proposed heuristic is 8 times repeated. In table 3 the comparison of computational efficiency is presented.

| Table 1. Value of $Sat(x_{opt})$ for different number of planning steps |
|-----------------------------|----------------|
| Planning step | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $Sat(x_{opt})$ | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1 |

| Table 2. Value of $U(x_{opt})$ |
|----------------------|---|
| Iteration | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $U$ | 0.1 | 0.1 | 0.2 | 0.33 | 0.44 | 0.61 | 0.80 | 1.00 |
| | 5 | 9 | 5 |

| Table 3. 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 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 4, 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)).

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).

Acknowledgement. This work has been funded from the Polish Ministry of Science and Higher Education funds in years 2008-2010.

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. So additional heuristic to receive the plan from the vector is needed.

Table 4. Solution for different methods

<table>
<thead>
<tr>
<th>step</th>
<th>action</th>
<th>LP</th>
<th>LP + heur.</th>
<th>Bin.int</th>
<th>Rounding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Move-to-table(A,B)</td>
<td>0.5</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Move-to-table(F,G)</td>
<td>0.5</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

References