Answer Set Planning under Action Costs*

Thomas Eiter¹, Wolfgang Faber¹, Nicola Leone², Gerald Pfeifer¹, and Axel Polleres¹

¹ Institut für Informationssysteme, TU Wien, A-1040 Wien, Austria
{eiter,faber,polleres}@kr.tuwien.ac.at
pfeifer@dbai.tuwien.ac.at

² Department of Mathematics, University of Calabria, I-87030 Rende (CS), Italy
leone@unical.it

Abstract. We present $K^C$, which extends the declarative planning language $K$ by action costs and optimal plans that minimize overall action costs (cheapest plans). As shown, this novel language allows for expressing some nontrivial planning tasks in an elegant way. Furthermore, it flexibly allows for representing planning problems under other optimality criteria as well, such as computing “fastest” plans (with the least number of steps), and refinement combinations of cheap and fast plans. Our experience is encouraging and supports the claim that answer set planning may be a valuable approach to advanced planning systems in which intricate planning tasks can be naturally specified and effectively solved.

1 Introduction

Recently, several declarative planning languages and formalisms have been introduced, which allow for an intuitive encoding of complex planning problems including ramifications, incomplete information, non-deterministic action effects, or parallel actions [13,18,17,19,12,4,5,6]. While these formalisms are designed to generate any plans that establish the planning goals, in practice we are usually interested in particular plans that are optimal with respect to an objective function which measures the quality (or cost) of a plan. Often, this is just the number of time steps to achieve the goal, and many systems are tailored to compute shortest plans (e.g. CMBP [4] and GPT [2] compute shortest sequential plans, while Graphplan [1] computes shortest parallel plans).

However, there are other important objective functions to consider. If executing an action (such as traveling from Vienna to Lamezia Terme) causes some cost, then we may desire a plan which minimizes the overall cost of the actions in that plan. In answer set planning [18], where plans are represented by answer sets of a logic program, this kind of problem has not been addressed so far, to the best of our knowledge.

In this paper, we address this issue and present an extension of the planning language $K$ [5,6], where one can associate actions with costs. The main contributions are:

- We define syntax and semantics of a modular extension to $K$. Costs are associated to an action by a designated where-clause describing a cost value.
- Action costs may be dynamic, as they potentially depend on the current stage of the plan when an action is considered for execution. Dynamic action costs have natural applications, such as variants of the well-known Traveling Salesperson example.

* This work was supported by FWF (Austrian Science Funds) under the projects P14781 and Z29-INF

© Springer-Verlag Berlin Heidelberg 2002
We sketch how planning with costs can be implemented by mapping it to answer set
programming, as realized in a system prototype that we have developed. The proto-
type, ready for experiments, is available at http://www.dlvsystem.com/K/.

Finally, we show that our language is capable of easily modeling optimal planning
under various criteria: computing (1) “cheapest” plans (which minimize overall
action costs); (2) “fastest” plans (with the least number of steps); and combinations
of these, viz. (3) fastest plans among the cheapest, and (4) cheapest plans among
the fastest. To our knowledge, task (3) has not been addressed in other works so far.

The extension of $\mathcal{K}$ by action costs provides a flexible tool for representing different
problems. Moreover, by $\mathcal{K}$’s nature, we get the possibility to easily combine dealing
with incomplete knowledge and plan quality, which is completely novel.

Our experience is encouraging and gives further evidence that answer set planning,
based on powerful logic programming engines, allows for the development of advanced
declarative planning systems in which intricate planning tasks can be naturally specified
and decently solved.

For space reasons, we provide proofs and more material in an extended paper [8].

\section{Review of Language $\mathcal{K}$}

In this section, we give a brief informal overview of the language $\mathcal{K}$. We assume
that the reader is familiar with action languages and the notions of actions, fluents, goals,
and plans and refer to [6] for further details. For illustration, we shall use the following
running example, for which a $\mathcal{K}$ encoding is shown in Figure I.

**Bridge crossing.** Four men want to cross a river at night. It is bridged by a plank bridge,
which can only hold up to two persons at a time. The men have a lamp, which must
be used in crossing, as it is pitch-dark and some planks are missing. The lamp must be
brought back; no tricks (like throwing the lamp or halfway crosses) are allowed.

A state in $\mathcal{K}$ is characterized by the truth values of fluents, describing relevant prop-
erties in the universe of discourse. A fluent may be true, false, or unknown in a state;
formally, a state is any consistent set $s$ of (possibly negated) legal fluent instances. Note
that in world-state planning, each fluent is either true or false in a state (this can be easily
emulated in $\mathcal{K}$).

An action is only applicable if some precondition holds in the current state, and its
execution may cause a modification of truth values of some fluents.

**Background Knowledge.** Static knowledge which is invariant over time is specified as a
disjunction-free Datalog program which we require to have a total well-founded model
(and therefore a unique answer set). In our example, the background knowledge is simply

\[
\text{person(joe). person(jack). person(will). person(ave).}
\]

\footnote{The well-founded model can be calculated in polynomial time and if it is total it corresponds
to the unique answer set.}