10.24425/acs.2021.137423

Archives of Control Sciences
Volume 31(LXVII), 2021
No. 2*, pages 375–399*

On transformation of conditional, conformant and parallel planning to linear programming

Adam GALUSZKA, Eryka PROBIERZ

Classical planning in Artificial Intelligence is a computationally expensive problem of finding a sequence of actions that transforms a given initial state of the problem to a desired goal situation. Lack of information about the initial state leads to conditional and conformant planning that is more difficult than classical one. A parallel plan is the plan in which some actions can be executed in parallel, usually leading to decrease of the plan execution time but increase of the difficulty of finding the plan. This paper is focused on three planning problems which are computationally difficult: conditional, conformant and parallel conformant. To avoid these difficulties a set of transformations to Linear Programming Problem (LPP), illustrated by examples, is proposed. The results show that solving LPP corresponding to the planning problem can be computationally easier than solving the planning problem by exploring the problem state space. The cost is that not always the LPP solution can be interpreted directly as a plan.

Key words: planning, conformant planning, conditional planning, parallel planning, uncertainty, linear programming, computational complexity

1. Introduction

Artificial Intelligence can be understood as 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 [20, 28]. Classical 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

A. Galuszka (corresponding author, e-mail: adam.galuszka@polsl.pl) and E. Probierz (e-mail: eryka.probierz@polsl.pl) are with Department of Automatic Control and Robotics, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland.

Received 29.03.2020. Revised 26.04.2021.

be created (e.g. [21]). 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. [5]). One can find examples of planning applications in manufacturing, production planning (e.g. [25]), logistics and agentics (eg. [11]).

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 [2, 4]. When all states of a planning problem (including an initial and a goal) are defined by a given set of conditions (also called predicates), then the problem is called STRIPS planning problem [23]. The planning systems have been successfully applied in planning modules of Deep Space One Spacecraft [31] and for elevators control in Rockefeler Center in New York [19]. One should mention STRIPS system is no longer used in its original form. The advanced version of STRIPS introduced in 1987 [24] is called Action Description Language (ADL) and other extensions of planning languages are standardized by Planning Domain Definition Language (PDDL) formalism ([15]. The latest PDDL extension is The Hierarchical Domain Definition Language (HDDL, [18]).

The problem becomes more complicated, if information about the modeled world is not sufficient to determine all facts necessary to describe an initial state of the world. Then, we say that the initial state of the problem is uncertain but can be represented by a set of possible initial states. A plan for solving such a problem may take the form of actions that are executed conditionally, based on new information emerging during the search for the plan. The inflow of new information is modeled by the so-called sensory actions, in such a way that the uncertainty of the information available is reduced by using information from the sensors. This approach is called conditional planning [30, 32].

In some cases, information from sensors may be unavailable e.g. sensors are damaged or broken down, receiving sensory information is too expensive or dangerous. Then, it is reasonable to search for a plan that is a solution to the planning problem independently of possible initial states. This approach is called conformant planning [30, 32]. Both conditional and conformant planning are more difficult to solve than a classical planning [17].

The cases, in which more than one action can be applied in one planning step, i.e. some actions can be performed simultaneously, constitute a large class of important planning problems. Such problem formulation allows to model multi-agent and multi-robot environments and is called parallel planning. Combining conformant and parallel planning leads to a problem, in which many agents interact in an uncertain environment with no possibility of performing sensing actions. Finding a solution to a parallel conformant planning problem is more difficult than for previous problems. To avoid this difficulty, in the paper we propose a heuristics for the transformation of the problem to a Linear Programming Problem (LPP), illustrated by examples.

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 [1] 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 [7]. The complexity of planning problems strongly depends on the complexity of the actions defined for the assumed domain (also [7]). 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. Planning in the presence of incompleteness is much more complicated [6] and belongs to the next level in the hierarchy of completeness. Precisely speaking, if the uncertainty about the initial state is modelled by a set of possible initial states, then planning problem is NP^{NP}-complete or Σ_2P-complete [3]. High level of computational complexity of planning causes that practical applications of planning under uncertainty are based on heuristic algorithms (e.g. [16, 27]. The newest approach provides a way to combine a method that does not explicitly consider any problem structure with techniques that do [33].

One of the heuristics is a transformation of planning to LPP. LPP formulations in classical planning are under newest investigations: in [29] the post-hoc optimization heuristic uses LPP to determine a real-valued factor for each heuristic in a set of pattern database abstractions, and the cost partitioning is derived by multiplying the costs of all operators that affect an abstraction with that factor. Many other heuristics can be expressed with an LPP over variables that express how often an operator is used [26]. The idea of representing STRIPS planning problems by linear constraints and objective function is also not new in the literature (see e.g. [22]). 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 [8] is to introduce additional linear constraints to LPP. It allows to solve optimally some class of classical planning problems using LP polynomial algorithms [9]. The cost is that not always the LP solution can be interpreted directly as a plan (what is followed by assumption $P \neq NP$). Also, the size of LPP increased (polynomially) very fast with the number of planning problem variables.

1.2. Contribution

In this paper a heuristic of the transformation of conditional, conformant and parallel planning problems to LPP, basing on extension of the transformation given in [12], is proposed. This is done because LPPs are known to be computational easy [9].

The following problems are presented and analyzed:

- transformation of conditional planning to Linear Programming,
- transformation of conformant planning to Linear Programming,
- transformation of parallel conformant planning to Linear Programming,
- computational complexity of solving transformed planning problems.

1.3. Organization of the paper

The paper is organized as follow: In section 2 conditional, conformant and parallel planning problems together with examples are introduced. In section 3 planning problems transformations are proposed. In section 4 exemplary transformed problems are solved. Remarks on computational complexity are given in section 5. All is concluded and future works are suggested in section 6.

2. Conditional, conformant and parallel planning problems

Following Bylander [7] it is assumed that planning problem Π consists of four sets $\Pi = \{C, O, I, G\}$:

- C is a finite set of *conditions*,
- O is a finite set of actions, where each action $o \in O$ takes the form $c^+, c^- \rightarrow c_+, c_-$, where:
  - $c^+ \subseteq C$ are so called *positive preconditions*,
  - $c^- \subseteq C$ are so called *negative preconditions*,
  - $c_+ \subseteq C$ are so called *positive postconditions*,
  - $c_- \subseteq C$ are so called *negative postconditions*,
- $I \subseteq C$ is an *initial state*,
- $G = \{G_+, G_-\}$ is a *goal situation*, where $G_+ \subseteq C$ are positive conditions (i.e. are true) and $G_- \subseteq C$ are negative conditions (i.e. are false).

In order to include the information that some conditions are unknown (assume k conditions can be true or false) in the description of the current problem state, one can introduce so called *k-states* proposed by [3]. In simple terms *k-state* is a pair (s, Σ), where s is the current problem state, and Σ is a set that consists of all possible initial states I. For unknown initial state set Σ consists of all states s, for which:

- condition $c \in C$ is true in the initial state (i.e. $c \in I$),
- condition $c \in C$ is false (i.e. $\neg c \in I$),
- if it is unknown whether condition $c \in C$ is true or false in the initial state then set Σ includes both states for which this condition is true and false.

The initial state I can be potentially any state from states included in set Σ. The number of possible initial states is denoted by w and is limited by k such that: $w \leqslant 2^k$. Such planning problem with incomplete information about initial state is called *conformant planning problem* and takes the form:

$$\Pi_{\text{conf}} = (C, O, \Sigma, G). \tag{1}$$

The result of applying action to the current state depends whether the action is ordinary or sensory. Description of this result is presented below, is based on [3] and is adopted to STRIPS problem.

For action o, *k-state* is described by a set $\{Result(S, \{o\}), Result(\Sigma, \{o\})\}$, where $Result(S, \{o\})$ is the same like in case with complete information, e.g.:

$$\begin{aligned}
&Result(S, \{\,\}) = S,\\
&Result(S, \{o\}) = \begin{cases} (S \cup c_+)\backslash c_- & \text{if } c^+ \subseteq S \wedge c^- \cap S = \emptyset;\\ S & \text{in opposite case,}\end{cases}\\
&Result(S, \{o_1, o_2, \ldots, o_n\}) = Result(Result(S, \{o_1\}),\ \{o_2, \ldots, o_n\}),
\end{aligned} \tag{2}$$

and:

$$Result(\Sigma, \{o\}) = \{Result(S', \{o\}) |\ S' \in \Sigma\}. \tag{3}$$

Modern intelligent systems are often equipped with sensors of different kind that are used to determine different properties of robot's environment. This information can be mapped to truth degree of conditions that define current problem state. Usually, it is done by introducing special actions called *sensory actions* [32]. As there is no formal extension of STRIPS planning by sensory actions, below the definition of these actions for k unknown conditions, as a special subset of STRIPS actions, is proposed.

Definition 1 *For k unknown conditions set of sensory actions O_s is a finite set of actions, where for each sensory action $o_s \in O_s$ it is needed to introduce two STRIPS sensory actions $\{o_s^t, o_s^f\} \in O_s$ that take the form:*

$o_s^t: c^+, c^- \rightarrow c_i$, if condition c_i is true after performing action o_s,
$o_s^f: c^+, c^- \rightarrow \neg c_i$, if condition c_i is false after performing action o_s,
$i = 1, 2, \dots, k$.

It follows that the maximal number of sensory actions $|O_s| = 2k$. The result of applying action to the current state depends on whether the action is ordinary or sensory. Such planning problem with incomplete information about initial state and sensory actions is called *conditional planning problem* and takes the form:

$$\Pi_{\text{cond}} = (C, (O, O_s), \Sigma, G). \tag{4}$$

The *plan* $\Delta_C = \langle o_1, o_2, \dots, o_n \rangle$ solves conformant planning problem if:

$$Result((S, \Sigma),\ , \Delta_C) = G. \tag{5}$$

Since all actions in Δ_C are ordered, Δ_C is called a *total order conformant plan*.

The *partial-order conformant plan* is denoted as $\Delta_{\text{POC}} = \{\Delta_{\text{SetC}}, \pi\}$, where $\Delta_{\text{SetC}} = \{o_1, o_2, \dots, o_n\}$ is the set of actions, and π is the non-returnable partial order defined on Δ_{SetC} (compare [2]). So, a *partial-order conformant plan* is a compact representation of a set of possible *total* ordered plans.

The *parallel partial-order conformant plan* is denoted as $\Delta_{\text{PPOC}} = \{\Delta_{\text{SetC}}, \pi, \#\}$, where $\{\Delta_{\text{SetC}}, \pi\}$ is Δ_{POC}, while # is a symmetrical relation, defined on the set Δ_{SetC}. $\# \subseteq (\pi \cup \pi^{-1})$ is called a non-concurrency relation and it indicates which actions cannot be applied in parallel.

The *plan* Δ_{COND} solves conditional planning problem if:

$$Result((S, \Sigma),\ , \Delta_{\text{COND}}) = G. \tag{6}$$

The plan consists of both classical ($o \in O$) and sensory actions ($o_s \in O_s$) and classical actions are performed conditionally and depend on the result of sensory action.

2.1. Example of conditional planning problem

Consider the problem of opening doors by the robot. It is assumed that robot can perform actions of pushing doors (*push_door*) and flipping lock (*flip_lock*). Additionally, it can perform sensory action of checking if doors are locked (*check_if_locked*). If doors are not locked ($\neg$*locked*) and not jammed ($\neg$*jammed*) then action *push_door* opens them. If doors are locked then action *push_door* jammed them. The goal is to open doors so condition *open* should be true in description of final problem state. Actions can be described as follow:

$$push\_door: \qquad \text{effect}: \ jammed \quad \text{if } \ locked, \tag{7}$$

$$\begin{array}{llll}
\mathit{push\_door}: & \text{effect}: \ \mathit{open} & \text{if } \neg \mathit{locked}, \neg \mathit{jammed}, & (8)\\
\mathit{flip\_lock}: & \text{effect}: \ \mathit{locked} & \text{if } \neg \mathit{locked}, & (9)\\
\mathit{flip\_lock}: & \text{effect}: \ \neg\mathit{locked} & \text{if } \mathit{locked}, & (10)\\
\mathit{check\_if\_locked}: & \text{determines} & \text{if } \mathit{locked} \text{ is } \mathit{true} \text{ or } \mathit{false}. & (11)
\end{array}$$

Action model in formulas (7)–(11) is different than classical cause-effect action. It is caused by the fact that actions effects in formulas (7)–(10) are formulated conditionally (action causes *set1_of_conditions* if *set2_of_conditions*) and action in formula (11) is formulated as a truth determination for unknown conditions). Both models can be translated into classical six cause-effect actions and take the form like in formulas (12)–(17):

$$o_1 = \mathit{push\_door}: \quad \mathit{locked} \rightarrow \mathit{jammed}, \tag{12}$$

$$o_2 = \mathit{push\_door\_1}: \quad \neg\mathit{locked}, \ \neg\mathit{jammed} \rightarrow \mathit{open}, \tag{13}$$

$$o_3 = \mathit{flip\_lock}: \quad \neg\mathit{locked} \rightarrow \mathit{locked}, \tag{14}$$

$$o_4 = \mathit{flip\_lock\_1}: \quad \mathit{locked} \rightarrow \neg\mathit{locked}, \tag{15}$$

$$o_5 = \mathit{check\_if\_locked}: \ \{\text{no preconditions}\} \rightarrow \mathit{locked}, \ \text{if } \mathit{locked} \text{ is } \mathit{true}, \tag{16}$$

$$o_6 = \mathit{check\_if\_locked\_1}: \{\text{no preconditions}\} \rightarrow \neg\mathit{locked}, \ \text{if } \mathit{locked} \text{ is } \mathit{false} \tag{17}$$

where $o_1, o_2, o_3, o_4 \in O$ and $o_5, o_6 \in O_s$.

Two separate actions are needed for one sensory action [32], so in the example $|O| = 6$. If the set of problem actions contains sensory actions it implies that the plan solving the problem is not a determined sequence of actions: if at least one action is sensory, then next actions depend on sensory action result. It leads to so called *conditional plans*. Assume in the example that in initial situation doors are closed and not jammed $\{\neg open, \neg jammed\}$, but it is unknown if they are locked. It leads to two possible initial problem states and theirs description is included in set Σ:

$$\Sigma = \{\{\neg open, \neg jammed, \ locked\}, \ \{\neg open, \ \neg jammed, \ \neg locked\}\}. \tag{18}$$

Remaining sets for STRIPS representation are:

$$C = \{c_1 = locked, \ \ c_2 = jammed, \ \ c_3 = open\}, \qquad I = \Sigma, \qquad G = \{open\}. \tag{19}$$

and conditional plan that solves the problem is:

$$\begin{aligned}\Delta = \{&\mathit{check\_if\_locked}, \ \ \text{if } \neg\mathit{locked} \text{ then } \mathit{push\_door}, \\ &\qquad\qquad \text{if } \mathit{locked} \text{ then } \mathit{flip\_lock}, \mathit{push\_door}\}.\end{aligned} \tag{20}$$

2.2. Example of conformant planning problem

To illustrate conformant planning consider the following simple model of the action of taking medication [32]:

$$Medicate: \textit{no preconditions, effects}: (when\ I \neg I)\ \ (when\ \neg H D) \tag{21}$$

in which I means the patient is Infected, H means he is Hydrated and D means Dangerous health state. If the patient is Infected before the action then he will no longer be Infected but if the patient takes medication when he is not Hydrated then the result is Dangerous health state.

Action model in formula (21) is different than classical one. It is caused by a fact that actions effects are formulated conditionally with the general schema: action causes *set1_of_conditions* if *set2_of_conditions*. Please note that such defined action has no preconditions but action effects are formulated conditionally. It implies that preconditions are indirectly defined by effects, so action Medicate is equivalent to four classical cause-effect actions:

$$\begin{array}{lll} Med_1: & preconditions: (I\ and\ H), & effects: \neg I, \\ Med_2: & preconditions: (I\ and\ \neg H), & effects: (\neg I\ and\ D), \\ Med_3: & preconditions: (\neg I\ and\ H), & effects: no\ effects, \\ Med_4: & preconditions: (\neg I\ and\ \neg H), & effects: D. \end{array} \tag{22}$$

As an example of conformant planning one can consider following problem Π_{Med} with two possible initial states:

$$\Pi_{Med} = \{C_{Med},\ O_{Med},\ \Sigma_{Med},\ G_{Med}\}, \tag{23}$$

where:

$$\begin{aligned} C_{Med} &= \{I,\ H,\ D\}, \\ O_{Med} &= \{Medicate: \textit{no preconditions, effects}: (when\ I \neg I)(when\ \neg H\ D); \\ &\qquad Drink: \textit{nopreconditions, effects}:\ H\}, \\ \Sigma_{Med} &= \{\{\neg I,\ \neg H,\ \neg D\},\ \ \{I,\ H,\ \neg D\}\}, \\ G_{Med} &= \{\neg I,\ \neg D\}. \end{aligned}$$

The plan that solves the problem cannot be just *Medicate* because we do not know whether the patient is hydrated or not. Conformant plan that solves the Π_{Med} problem is:

$$\Delta = \{Drink,\ Medicate\}. \tag{24}$$

2.3. Example of parallel conformant planning problem

To illustrate a parallel conformant planning problem consider the following simple *bomb in the toilet* problem with one action containing conditional effects:

$$\begin{aligned} Dunk(P): preconditions:\ & package(P),\ bomb(B), \\ & effects:\ if\ in(P,\ B)\ then\ defused(B), \end{aligned} \tag{25}$$

meaning that if there is a bomb B and there is a package P then action *dunk* causes that bomb B is defused if it was in package P. So if there is no bomb in package, the action *dunk* has no effects.

Action model in formula (25) is different than classical action. It is caused by a fact that actions effects are formulated conditionally with general schema: action causes *set1_of_conditions* if *set2_of_conditions*. Please note that an action defined in such way has no preconditions but action effects are formulated conditionally. It implies that preconditions are indirectly defined by effects, so action *Dunk* is equivalent to two classical actions:

$$\begin{aligned} Dunk_1(P):\quad & preconditions: package(P),\ bomb(B),\ in(P,\ B), \\ & effects:\quad defused(B), \\ Dunk_2(P):\quad & preconditions: package(P),\ bomb(B),\ not(in(P,\ B)), \\ & effects:\quad no\ effects. \end{aligned} \tag{26}$$

Now, let us consider the following problem Π_{BT} with two possible initial states (bomb is in package 1 or in package 2):

$$\Pi_{BT} = \{C_{BT},\ O_{BT},\ \Sigma_{BT},\ G_{BT}\}, \tag{27}$$

where:

$$\begin{aligned} C_{BT} &= \{package(P1), package(P2), bomb(B), in(P1, B), in(P2, B), defused(B)\}, \\ O_{BT} &= \{Dunk\}, \\ \Sigma_{BT} &= \{\{package(P1),\ package(P2),\ bomb(B),\ in(P1, B)\}, \\ &\qquad \{package(P1),\ package(P2),\ bomb(B),\ in(P2, B)\}\}, \\ G_{BT} &= \{defused(B)\}. \end{aligned}$$

The conformant plan that solves the Π_{BT} problem is:

$$\begin{aligned} \Delta_{\mathrm{CBT}} &= \langle Dunk(P1),\ Dunk(P2)\rangle \qquad \text{or} \\ \Delta_{\mathrm{CBT}} &= \langle Dunk(P2),\ Dunk(P1)\rangle. \end{aligned} \tag{28}$$

The partial-order conformant plan that solves the Π_{BT} problem is:

$$\Delta_{\mathrm{POCBT}} = \{Dunk(P2),\ Dunk(P1)\}.(29) \tag{29}$$

If actions in Δ_{POCBT} can be performed in parallel ($\#_{BT} = \emptyset$), then $\Delta_{\mathrm{POCBT}} = \Delta_{\mathrm{PPOCBT}}$ and the problem is solved in one step.

3. Transformation to LP

Following [8], 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.

Assume that if $c \in C$ is a condition and if the planning process is divided into l steps and i is the step index ($i = 0, 1, \ldots, l$) then $(l+1)$ variables are needed for this condition: $c(0), c(1), \ldots, c(l)$. If in turn $o \in O$ is an action, then we need l variables for each action: $o(0), o(1), \ldots, o(l-1)$. If $o_s \in O_s$ is a sensory action, then we need $2l$ variables for each sensory action: $o_s^t(0), o_s^t(1), \ldots, o_s^t(l-1)$, and $o_s^f(0), o_s^f(1), \ldots, o_s^f(l-1)$. Thus, the arguments for conditions c and operators o are extended by the index of the planning step.

The goal function reaches its maximum value if the value of the variable $c(l)$ corresponding to condition is: $c(l) = 1$ if $c \in G_+$ and $c(l) = 0$ if $c \in G_-$, i.e. the goal state is true in the last step of planning l.

Basing on above one can build LPP having vector of decision variables x:

$$x = \{c(0),\ o(0),\ o_s^t(0),\ o_s^f(0),\ c(1),\ o(1),\ o_s^t(1),\ o_s^f(1),\ \ldots,\\ c(l-1),\ o(l-1),\ o_s^t(l-1),\ o_s^f(l-1),\ c(l)\}.$$

Assume now that the set $G_+ = \{c_1^{\text{pos}},\ c_2^{\text{pos}},\ \ldots,\ c_n^{\text{pos}}\}$ and the set $G_- = \{c_1^{\text{neg}}, c_2^{\text{neg}}, \ldots, c_m^{\text{neg}}\}$, i.e. the goal consists of n positive and m negative conditions, so there are $(n+m)$ variables that constitutes LP objective function:

$$c^{\text{pos}}(l) = \left[c_1^{\text{pos}}(l),\ c_2^{\text{pos}}(l),\ \ldots,\ c_n^{\text{pos}}(l)\right];$$
$$c^{\text{neg}}(l) = \left[c_1^{\text{neg}}(l),\ c_2^{\text{neg}}(l),\ \ldots,\ c_m^{\text{neg}}(l)\right].$$

The objective function to be maximized is:

$$\text{Max} \leftarrow f((c^{\text{pos}}(l),\ c^{\text{neg}}(l)) = \left[\sum_{i=1}^{n}\left(c_i^{\text{pos}}(l)\right) + \sum_{j=1}^{m}\left(1 - c_j^{\text{neg}}(l)\right)\right]. \tag{30}$$

Since each component of (30) should be equal to 1 (one) if the goal is achieved, the optimal value of objective function (f_{opt}) to be maximized is known prior to solving LP problem and is $f_{\text{opt}} = (n + m)$. The optimization problem is formulated as:

Find minimal number of steps “l” for which $f_{\text{opt}} = (n + m)$.

Inequality constraints (of the form “greater or equal to”) express the property that actions can be applied if their preconditions are true. The left side of the inequality consists of variable corresponding to the precondition. If precondition is not satisfied then variable value for this condition is ‘0’. The right side of inequality is the sum of variables corresponding to all actions having the precondition. Thus, if left side is ‘0’ any action having the precondition cannot be applied. Dependently of the initial state representation and the number of actions taken in parallel inequalities differ in detail.

Equality constraints describe changes of variables value for conditions due to action application. Dependently of the initial state representation and the number of actions taken in parallel these equalities also differ in detail.

To model uncertainty about *truth* or *false* of unknown condition c it is proposed to use three-valued Kleen’s logic system. In this system logic values of condition are mapped into set $\left\{0,\ \frac{1}{2},\ 1\right\}$. Sentences $T(a) = 0$, $T(a) = 1$ and $T(a) = \frac{1}{2}$ denote that condition “a” is false, truth or that nothing can be said about truthfulness of “a” [10]. Following subsections introduce constraints dependently on the planning problem.

3.1. Conditional planning problem

For conditional planning problem (4) with two classical actions and one sensory action that checks the unknown c_3:

$$\begin{aligned} o_1 &: \quad c_1 \rightarrow c_2\,, \\ o_2 &: \quad \neg c_1,\ \neg c_2 \rightarrow \neg c_3\,, \\ o_{s1}^t &: \quad \{\ \} \rightarrow c_3\,, \\ o_{s1}^f &: \quad \{\ \} \rightarrow \neg c_3\,, \end{aligned} \tag{31}$$

the set of inequalities is:

$$\begin{aligned} c_1(i) &\geqslant o_1(i), \\ 1 - c_1(i) &\geqslant o_2(i), \\ 1 - c_2(i) &\geqslant o_2(i), \end{aligned} \tag{32}$$

and set of equalities is:

$$\begin{aligned} c_2(i+1) &= c_2(i) + o_1(i), \\ c_3(i+1) &= c_3(i) - o_2(i) + \frac{1}{2}o_{s1}^t(i) - \frac{1}{2}o_{s1}^f(i). \end{aligned} \tag{33}$$

One should note that the value c_3 in next planning step, $c_3(i+1)$, can be modified in two ways:

1) if c_3 is true in current planning problem state (i.e. $c_3(i) = 1$) than it can stay false in next step after applying action o_2;

2) if c_3 is unknown in current planning problem state (i.e. $c_3(i) = \frac{1}{2}$) than it can stay true or false in next step dependently of the sensory action result.

3.2. Conformant planning problem

In conformant planning there are no sensory actions, so for conformant planning problem (1) and two actions with unknown c_3:

$$\begin{aligned} o_1 :& \quad c_1 \to c_2\,, \\ o_2 :& \quad \neg c_1,\ \neg c_2 \to \neg c_3\,, \end{aligned}$$

there are two possible worlds: the first one in which c_3 is true and the second one in which c_3 is false. The solution is the plan that solves the problem independently of the world we are in. The set of inequalities is:

$$\begin{aligned} c_1^w(i) &\geqslant o_1(i), \\ 1 - c_1^w(i) &\geqslant o_2(i), \\ 1 - c_2^w(i) &\geqslant o_2(i), \end{aligned} \tag{34}$$

and set of equalities is:

$$\begin{aligned} c_2^w(i+1) &= c_2^w(i) + o_1(i), \\ c_3^w(i+1) &= c_3^w(i) - o_2(i), \end{aligned} \tag{35}$$

where $w = 1, 2$.

3.3. Parallel conformant planning problem

For parallel conformant planning problem (1) there is a possibility to perform many actions at the same planning step. One should note that actions o_1 and o_2 in (31) cannot be taken at the same step since their preconditions are mutually excluding. Assuming that r is the maximal number of actions performed in parallel one finds modifications in inequality (34):

$$r * c_1^w(i) \geqslant o_1(i).$$

4. Exemplary results

Below transformation results for examples introduced in sections 2.1, 2.2 and 2.3 are shown.

4.1. Transformation of conditional planning problem

If l is the number of planning steps then variables for conditions (19) are:

$$\begin{aligned} c_1(i) &= locked(i), & c_2(i) &= jammed(i), \\ c_3(i) &= open(i), & i &= 0, 1, \ldots, l, \end{aligned} \tag{36}$$

for actions (12)–(17):

$$\begin{aligned} o_1(i) &= push\_door(i), \\ o_2(i) &= push\_door\_1(i), \\ o_3(i) &= flip\_lock(i), \\ o_4(i) &= flip\_lock\_1(i), \\ o_5(i) &= check\_if\_locked(i), \\ o_6(i) &= check\_if\_locked\_1(i), \\ i &= 0, 1, \ldots l{-}1. \end{aligned} \tag{37}$$

The description of initial state can be transformed from the set of possible initial states Σ to set of equality constraints with variable values for unknown conditions equal to $\frac{1}{2}$:

$$\begin{aligned} c_1(0) &= locked(0) = 0.5, \\ c_2(0) &= jammed(0) = 0, \\ c_3(0) &= open(0) = 0. \end{aligned} \tag{38}$$

Goal state is reached if condition open is true in last planning step, so the objective function of LP is:

$$\max \leftarrow c_3(l) = open(l). \tag{39}$$

The planning problem is solved if the optimal value of f is equal to 1, meaning that one conditions is true. It leads to following formulation of optimization problem: *Find minimal number of planning steps l, such that* $f = 1$.

The set of constraints is:

– one action in one planning step (these are equality constraints):

$$\begin{aligned} &push\_door(i) + push\_door\_1(i) + flip\_lock(i) + flip\_lock\_1(i) + \\ &+ check\_if\_locked(i) + check\_if\_locked\_1(i) = 1, \quad i = 0, 1, 2, \ldots l{-}1, \end{aligned} \tag{40}$$

– actions can be applied if preconditions are true (these are inequality constraints):

$$\begin{aligned} locked(i) &\geqslant push\_door(i) + flip\_locked\_1(i), \\ 1 - locked(i) &\geqslant push\_door\_1(i) + flip\_lock(i), \\ 1 - jammed(i) &\geqslant push\_door\_1(i), \quad i = 0, 1, 2, \ldots, l{-}1, \end{aligned} \tag{41}$$

– changes of variables for conditions due to action application (these are equality constraints):

$$\begin{aligned} jammed(i+1) &= jammed(i) + push\_door(i), \\ open(i+1) &= open(i) + push\_door\_1(i), \quad i = 0, 1, 2, \ldots, l{-}1. \end{aligned} \tag{42}$$

Constraints (41) and (42) are only formulas for actions (12)–(15). Now it is necessary to model the influence of sensory actions that will change the value of variable for uncertain condition $locked = \frac{1}{2}$:

$$\begin{aligned} locked(i+1) = locked(i) &+ 0.5check\_if\_locked(i) \\ &- 0.5check\_if\_locked\_1(i). \end{aligned} \tag{43}$$

Initially $locked(0) = \frac{1}{2}$. The condition locked can be determined by sensory action *check_if_locked* modeled by 2 actions $o_5 = check\_if\_locked$ that determines *locked* and $o_6 = check\_if\_locked\_1$ that determines $\neg\, locked$. Thus, the variable value of the action o_5 should increase the variable value of the condition $locked(i)$ from "$\frac{1}{2}$" to "1". Similarly, the variable value of the action o_6 should decrease the variable value of the condition $locked(i)$ from "$\frac{1}{2}$" to "0", what is expressed by (43). It should be noted that that other classical actions also influences the value of $locked(i)$. The variable of action $o_3(i) = flip\_lock(i)$ changes the $locked(i)$ value to "1" if it was "0". The variable of action $o_4(i) = flip\_lock\_1(i)$ changes the $locked(i)$ value to "0" if it was "1". Thus, the constraint (43) should also model these changes. It leads to:

$$\begin{aligned} locked(i+1) = locked(i) &+ 0.5check\_if\_locked(i) \\ &- 0.5check\_if\_locked\_1(i) + flip\_lock(i) - flip\_lock\_1(i). \end{aligned}$$

Last constraints are implications of the fact that sensory actions for each unknown condition are performed only once during planning process:

$$\Sigma check\_if\_locked(i) + \Sigma check\_if\_locked\_1(i) = 1.$$

Optimal solution *xopt* depends on the result of application of sensory action *check_if_locked*. If the door were locked, it would correspond to effects of action o_5, and it is modeled by introducing additional equality constraint to LP:

$$o_5(0) = check\_if\_locked(0) = 1, \tag{44}$$

and if the door were $\neg locked$, it corresponds to effects of action o_6, and is modeled by introducing additional equality constraint to LP:

$$o_6(0) = check\_if\_locked\_1(0) = 1. \tag{45}$$

In case (44) the number of planning steps that satisfies goal is $l = 3$ and the vector *xopt* that maximizes (39) can be directly interpreted as a plan: $\Delta_1 = \{o_5, o_4, o_2\} = \{check\_if\_locked, flip\_lock\_1, push\_door\_1\}$. In the opposite case, one should apply additional heuristics or methods that leads to binary integer solution (see e.g. [12]). In case (45) the number of planning steps that satisfies goal is $l = 2$ and vector *xopt* that maximizes (39) can be directly interpreted as a plan: $\Delta_2 = \{o_6, o_2\} = \{check\_if\_locked\_1, push\_door\_1\}$. Plans Δ_1 and Δ_2 correspond to conditional plan (7).

4.2. Transformation of conformant planning problem

If w is the index of possible initial world state, then variables of the problem (23) for conditions are:

$$\begin{gathered} c_1^w(i) = I^w(i), \quad c_2^w(i) = H^w(i), \quad c_3^w(i) = D^w(i), \\ i = 0, 1, \ldots, l, \quad w = 1, 2, \end{gathered} \tag{46}$$

for actions:

$$\begin{aligned} o_1(i) &= Med_1(i), \\ o_2(i) &= Med_2(i), \\ o_3(i) &= Med_3(i), \\ o_4(i) &= Med_4(i), \\ o_5(i) &= Drink(i), \quad i = 0, 1, \ldots, l-1. \end{aligned} \tag{47}$$

The initial state is a disjunction of two possibilities. It is modelled by a set of equality constraints:

$$\begin{aligned} &c_1^1(0) = I^1(0) = 0, && c_2^1(0) = H^1(0) = 0, \\ &c_3^1(0) = D^1(0) = 0, \\ &c_1^2(0) = I^2(0) = 1, && c_2^2(0) = H^2(0) = 1, \\ &c_3^2(0) = D^2(0) = 0, \end{aligned} \tag{48}$$

Goal state $G_{Med} = \{\neg I, \neg D\}$ is reached if conditions I and D are false in last planning step in each world, so the objective function of LP, mapping ¬c into variable value in step i equal to $(1 - c(i))$, is:

$$\max \leftarrow f = \left(1 - c_1^1(l)\right) + \left(1 - c_3^1(l)\right) + \left(1 - c_1^2(l)\right) + \left(1 - c_3^2(l)\right). \tag{49}$$

The planning problem is solved if the optimal value of f is equal to 4, meaning two conditions in both worlds are false. It leads to following formulation of optimization problem: *Find minimal number of planning steps l, such that* $f = 4$.

Set of constraints is:

– one action in one planning step (these are equality constraints):

$$\begin{aligned} &\sum_{k=1}^{n} o_k(i) = 1, \\ &\sum_{k=1}^{n} o_k(i) = 1, \quad i = 0, 1, 2, \ldots l{-}1, \end{aligned} \tag{50}$$

– actions can be applied if preconditions are true (these are inequality constraints):

$$\begin{aligned} I^w(i) &\geqslant Med_1(i) + Med_2(i), \\ (1 - I^w(i)) &\geqslant Med_3(i) + Med_4(i), \\ H^w(i) &\geqslant Med_1(i) + Med_3(i), \\ (1 - H^w(i)) &\geqslant Med_2(i) + Med_4(i), \\ &i = 0, 1, 2, \ldots l{-}1, \quad w = 1, 2; \end{aligned} \tag{51}$$

– changes of variables for conditions due to action application (these are also equality constraints):

$$\begin{aligned} I^w(i+1) &= I^w(i) - Med_1(i) - Med_2(i), \\ D^w(i+1) &= D^w(i) + Med_2(i) + Med_4(i), \\ H^w(i+1) &= H^w(i) + Drink(i), \\ &i = 0, 1, 2, \ldots l{-}1, \quad w = 1, 2. \end{aligned} \tag{52}$$

Last equality constraint in (52) should be studied more carefully. First action in plan (4) *Drink* applied to possible initial state when patient is hydrated leads to infeasible value of variable $H^2(1)$:

$$H^2(1) = H^2(0) + Drink(0) = 1 + 1 = 2,$$

so one should introduce additional balancing variable for each condition in each planning step to avoid infeasibility:

$$\begin{aligned} I^w(i+1) + I_b^w(i+1) &= I^w(i) - Med_1(i) - Med_2(i), \\ D^w(i+1) + D_b^w(i+1) &= D^w(i) + Med_2(i) + Med_4(i), \\ H^w(i+1) + H_b^w(i+1) &= H^w(i) + Drink(i), \\ &i = 0, 1, 2, \ldots l{-}1, \quad w = 1, 2. \end{aligned} \tag{53}$$

Basing on formulas (46) to (53) it is easy derive general formulas for any problem (1). Table 1 presents the optimal solution of (30) which is divided into sections that correspond to variable values of actions o_i and conditions c_i, where i is the index of planning step.

Table 1: Optimal solution *xopt* for transformation of the conformant planning problem example

world	variable	$i = 0$	$i = 1$	$i = 2$	add1	add2
$w1$	I	0	0	0	0	0
	H	0	1	1	0	0
	D	0	0	0	0	0
$w2$	I	1	1	0	0	0
	H	1	1	1	1	0
	D	0	0	0	0	0
O	med1	0	0	–	–	–
	med2	0	0	–	–	–
	med3	0	1	–	–	–
	med4	0	0	–	–	–
	drink	1	0	–	–	–

It should be noted that values of variables for actions are binary integer, so the solution presented in Table 1 can be directly interpreted as a plan: $\Delta = \{Drink,\ Medicate\}$. In the opposite case, one should apply additional heuristics or methods that leads to binary integer solution (see e.g. [12]).

4.3. Transformation of parallel conformant planning problem

If l is the number of planning steps and w is the number of possible initial world states then variables of the problem (27) for conditions are [14]:

$$\begin{aligned} &c_1(i) = package(P1, i), \quad c_2(i) = package(P2, i), \quad c_3(i) = bomb(B, i),\\ &c_4^w(i) = in(P1, B, i)^w, \quad c_5^w(i) = in(P2, B, i)^w, \quad c_6^w(i) = defused(B, i)^w,\\ &i = 0, 1, \ldots, l, \quad w = 1, 2, \end{aligned} \tag{54}$$

for actions:

$$\begin{aligned} o_1^w(i) = Dunk_1(P1, i)^w, \quad o_2^w(i) = Dunk_2(P1, i)^w,\\ o_3^w(i) = Dunk_1(P2, i)^w, \quad o_4^w(i) = Dunk_2(P2, i)^w,\\ i = 0, 1, \ldots, l{-}1, \quad w = 1, 2. \end{aligned} \tag{55}$$

The initial state is a disjunction of two possibilities. It is modelled by a set of equality constraints:

$$\begin{aligned} &package(P1,0) = 1, \quad package(P2,0) = 1, \quad bomb(B,0) = 1,\\ &in(P1,B,0)^1 = 1, \quad in(P2,B,0)^1 = 0, \quad defused(B,0)^1 = 0,\\ &in(P1,B,0)^2 = 0, \quad in(P2,B,0)^2 = 1, \quad defused(B,0)^2 = 0.(56) \end{aligned} \tag{56}$$

Goal state G_{BT} is reached if condition (*defused* B) is *true* in last planning step in each world, so the objective function of LP is:

$$\max \leftarrow f = defused(B,l)^1 + defused(B,l)^2. \tag{57}$$

It leads to following formulation of optimization problem: *Find minimal number of planning steps l, such that $f = 2$.*

The set of constraints is given by:

– actions can be applied if preconditions are true (these are inequality constraints), so we have:

$$\begin{aligned} package(P1,i) &\geqslant Dunk_1(P1,i)^w + Dunk_2(P1,i)^w,\\ package(P2,i) &\geqslant Dunk_1(P2,i)^w + Dunk_2(P2,i)^w,\\ r * bomb(B,i) &\geqslant Dunk_1(P1,i)^w + Dunk_2(P1,i)^w + Dunk_1(P2,i)^w\\ &\quad + Dunk_2(P2,i)^w,\\ in(P1,B,i)^w &\geqslant Dunk_1(P1,i)^w,\\ in(P2,B,i)^w &\geqslant Dunk_1(P2,i)^w,\\ (1 - in(P1,B,i)^w) &\geqslant Dunk_2(P1,i)^w,\\ (1 - in(P2,B,i)^w) &\geqslant Dunk_2(P2,i)^w, \quad i = 0,1,2,\ldots,l{-}1, \quad w = 1,2, \end{aligned} \tag{58}$$

where r is a natural number indicating how many actions can be performed in parallel (in our example $r = 2$),

– changes of variables for conditions due to action application (these are equality constraints), so we have:

$$\begin{aligned} defused(B,i{+}1)^w = defused(B,i)^w + Dunk_1(P1,i)^w + Dunk_1(P2,i)^w,\\ i = 0,1,2,\ldots,l{-}1, \quad w = 1,2. \end{aligned} \tag{59}$$

The equality constraint (59) should be studied more carefully. If actions $Dunk_1(P1,i)^w$ and $Dunk_1(P2,i)^w$ are applied in parallel in the same planning step, then the value of condition $defused(B,i{+}1)^w$ becomes infeasible. In this case, one should introduce an additional balancing variable for each condition in each planning step to avoid infeasibility:

$$\begin{aligned} defused(B,i{+}1)^w + defused(B,i{+}1)_b{}^w = defused(B,i)^w +\\ +Dunk_1(P1,i)^w + Dunk_1(P2,i)^w,\\ i = 0,1,2,\ldots,l{-}1, \qquad w = 1,2. \end{aligned} \tag{60}$$

Table 2 presents the optimal solution of the parallel conformant planning problem as well as the two additional test problems with possible initial states given by set of equalities (61) and (62):

$$\begin{array}{rrr} package(P1,0) = 1, & package(P2,0) = 1, & bomb(B,0) = 1, \\ in(P1,B,0)^1 = 1, & in(P2,B,0)^1 = 0, & defused(B,0)^1 = 0, \\ in(P1,B,0)^2 = 1, & in(P2,B,0)^2 = 1, & defused(B,0)^2 = 0; \end{array} \tag{61}$$

$$\begin{array}{rrr} package(P1,0) = 1, & package(P2,0) = 1, & bomb(B,0) = 1, \\ in(P1,B,0)^1 = 1, & in(P2,B,0)^1 = 0, & defused(B,0)^1 = 0, \\ in(P1,B,0)^2 = 0, & in(P2,B,0)^2 = 0, & defused(B,0)^2 = 0.(62) \end{array} \tag{62}$$

Table 2: Optimal solution *xopt* for the parallel conformant planning problem (56) as well as (61) and (62) problems

world	LP variable	problem (56)	problem (61)	problem (62)
both	package(P1,0)	1	1	1
	package(p2,0)	1	1	1
	bomb(B,0)	1	1	1
world 1	in(P1,B,0)	1	1	1
	in(P2,B,0)	0	0	0
	defused(B,0)	0	0	0
world 2	in(P1,B,0)	0	1	0
	in(P2,B,0)	1	1	0
	defused(B,0)	0	0	0
world 1	Dunk1(P1,0)	1	1	1
	Dunk2(P1,0)	0	0	0
	Dunk1(P2,0)	0	0	0
	Dunk2(P2,0)	0	0	0
world 2	Dunk1(P1,0)	0	1	0
	Dunk2(P1,0)	0	0	0
	Dunk1(P2,0)	1	0	0
	Dunk2(P2,0)	0	0	0
world 1	defused(B,1)	1	1	1
world 2	defused(B,1)	1	1	0
–	objective f	2	2	1

In the first one (61) there is a bomb in first package but it is uncertain whether it is in the second package, in second one (62) there is no bomb in second package but it is uncertain whether it is in first package.

It should be noted that values of variables for actions are binary integer, so the solution presented in Table 1 can be directly interpreted as a plan:

$$\Delta_{\text{PPOCBT}} = \{Dunk(P2),\ Dunk(P1)\}.$$

In the opposite case, one should apply additional heuristics or methods that lead to a binary integer solution.

5. Remarks on computational complexity of transformed planning problems

Let us introduce complexity classes $\boldsymbol{P}$ and $\Sigma_k\boldsymbol{P}$. Following [3], a decision problem is a problem of determining whether a given input w satisfies a certain property F (i.e., in set-theoretic terms, whether it belongs to the corresponding set $S = \{w|F(w)\}$). For every positive integer k, a problem belongs to the class $\Sigma_k\boldsymbol{P}$ if the formula $F(w)$ can be represented as:

$$\exists u_1\ \forall u_2\ \ldots\ F(u_1,\ u_2,\ \ldots,\ u_n,\ w),$$

where $F(u_1, u_2, \ldots, u_n, w)$ is a tractable property, and all k quantifiers run over words of tractable length (i.e., of length limited by some given polynomial of the length of the input).

5.1. Complexity of transformed conformant planning problem

Basing on above notation one can represent formula (1) for conformant planning as:

$$\exists\Delta\ \forall I\Pi\,(\Delta, \Pi(C, O, I, G))\,, \tag{63}$$

where the initial state I can be potentially any state from states included in the set Σ. It follows that conformant planning is in $\Sigma_2\boldsymbol{P}$. It is also a complete problem [3].

The complexity of the heuristic presented in the paper results from the size of LP problem, i.e. the number of variables and constraints for the problem (1). The number of variables depends on the number of conditions, actions and planning steps is:

$$p = w|C|(l+1) + w|O|l = p_1 + p_2\,, \tag{64}$$

where:

$p_1 = w|C|(l+1)$ – the number of variables corresponding to conditions,
$p_2 = w|O|l$ – the number of variables corresponding to actions.

The number of constraints is:

- $w|C|$ equality constraints to define the initial state, since the number of constraints needed to define the initial state for each belief state is $|C|$,
- $|C|l$ equality constraints to define the change of variable values after performing the action,
- $|C|l$ inequality constraints to define actions preconditions,
- $2p$ inequality constraints for variable values $\langle 0, 1\rangle$.

In a general case, for problems with the number of variables and constraints limited polynomially by the size of the planning problem, it can be shown [12] that transformation of planning to LP takes time: $\boldsymbol{T} = \boldsymbol{O}(nl)$, where n is the size of the problem, $n = (w|C| + w|O|)$. If, additionally, it is assumed that the number of planning steps does not increase exponentially with the size of the problem, then transformation of planning to LP is polynomial with complexity $\boldsymbol{T} = \boldsymbol{O}(n^3)$. The heuristics of the transformation of planning with incomplete information about initial state and determined effect of actions to LP has two properties:

a) one should introduce an additional balancing variable for each condition in each planning step to avoid possible infeasibility of variable values,

b) given any feasible solution of LP problem x connected with planning problem $\Pi = (C, O, \Sigma, G)$, it is easy to check (in polynomial time) whether the solution corresponds to plan that solves Π.

From the property a) it follows that polynomial time, depending on the problem size, is needed to solve LP problems that represents incomplete planning: $\boldsymbol{T} = \boldsymbol{O}(n^3)$. From the property b) it follows that the heuristic is in NP.

5.2. Complexity of transformed conditional planning problem

In general case, for STRIPS problems with number of variables and constraints limited polynomially by the size of planning problem, it can be shown [12] that transformation of planning to LP takes time:

$$\boldsymbol{T} = \boldsymbol{O}\left((|C| + |O|)l\right). \tag{65}$$

If additionally it is assumed that the number of planning steps does not increase exponentially with the size of the problem, then transformation of planning to LP is polynomial with complexity $\boldsymbol{T} = \boldsymbol{O}(n^3)$, where n is the size of the problem, $n = (|C| + |O|)$. The heuristic of the transformation of planning with incomplete information about initial state and determined effect of actions to LP has two properties:

c) for k sensory actions one should perform 2^k transformations to LP,

d) given any admissible solution of LP problem x connected with planning problem $\Pi = \{C, O, \Sigma, G\}$ it is easy to check (in polynomial time) whether the solution corresponds to plan that solves Π.

The property a) follows that it is needed polynomial time depended on problem size an exponential time depended on the number of sensory actions to solve LP problems that represents incomplete planning: $\boldsymbol{T} = \boldsymbol{O}(n^3 2^k)$. The property b) follows that heuristic is in NP.

6. Conclusion

In this paper the transformation and the computational complexity of conditional planning, conformant planning and parallel conformant planning problems to LP were presented and analyzed.

Important planning problems are those where more than one agent interacts with the problem environment simultaneously. They arise in multi-agent and multi-robot environments. Additionally, it is assumed here that maximal number of actions applied to current problem state is r. It can occur when r agents act on the same problem state or one agent is able to perform r actions at a time. It should be noted that in real-life problems application of an action to a problem state does not always lead to expected effects. It is particularly important in cases where action outcomes are uncertain, as well, and when a condition that is determined can become undetermined. Future works will be focused on introducing uncertain action outcomes to LP transformation.

Acknowledgement

The work of Adam Galuszka was supported by the SUT grant No 02/060/RGP20/0019. The work of Eryka Probierz was supported in part by the European Union through the European Social Fund as a scholarship under Grant POWR.03.02.00-00-I029, and in part by the Silesian University of Technology (SUT) through the subsidy for maintaining and developing the research potential grant in 2021 for young researchers in analysis. This work was supported by Upper Silesian Centre for Computational Science and Engineering (GeCONiI) through The National Centre for Research and Development (NCBiR) under Grant POIG.02.03.01-24-099/13. Partial results of the first author have been presented on conferences: Methods and Models in Automation and Robotics in 2015 and European Simulation Multiconference in 2018.

References

[1] J.L. Ambite and C.A. Knoblock: Planning by rewriting. *Journal of Artificial Intelligence Research*, **15** (2001), 207–261, DOI: 10.1613/jair.754.

[2] Ch. Backstrom: Computational Aspects of Reordering Plans. *Journal of Artificial Intelligence Research*, **9** (1998), 99–137, DOI: 10.1613/jair.477.

[3] 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, DOI: 10.1007/3-540-44957-4_59.

[4] R. Bartak: Constraint satisfaction techniques in planning and scheduling: An introduction. *Archives of Control Sciences*, **18**(2), (2008), DOI: 10.1007/s10845-008-0203-4.

[5] A. Bhattacharya and P. Vasant: Soft-sensing of level of satisfaction in TOC product-mix decision heuristic using robust fuzzy-LP, *European Journal of Operational Research*, **177**(1), (2007), 55–70, DOI: 10.1016/j.ejor.2005.11.017.

[6] J. Blythe: An Overview of Planning Under Uncertainty. Pre-print from *AI Magazine*, **20**(2), (1999), 37–54, DOI: 10.1007/3-540-48317-9_4.

[7] T. Bylander: The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence*, **69** (1994), 165–204, DOI: 10.1016/0004-3702(94)90081-7.

[8] T. Bylander: A Linear Programming Heuristic for Optimal Planning. In Proc. of *AAAI Nat. Conf.*, (1997).

[9] L.G. Chaczijan: A polynomial algorithm for linear programming. *Dokł. Akad. Nauk SSSR*, 244 (1979), 1093–1096.

[10] E.R. Dougherty and Ch.R. Giardina: *Mathematical Methods for Artificial Intelligence and Autonomous Systems*, Prentice-Hall International, Inc. USA, 1988.

[11] I. Elamvazuthi, P. Vasant, and T. Ganesan: Fuzzy Linear Programming using Modified Logistic Membership Function, *International Review of Automatic Control*, **3**(4), (2010), 370–377, DOI: 10.3923/jeasci.2010.239.245.

[12] A. Galuszka: On transformation of STRIPS planning to linear programming. *Archives of Control Sciences*, 3 (2011), 227–251, DOI: 10.2478/v10170-010-0042-3.

[13] A. Galuszka, W. Ilewicz, and A. Olczyk: On Translation of Conformant Action Planning to Linear Programming. *Proc. 20th International Conference on Methods and Models in Automation & Robotics*, 24–27 August, (2005), 353–357, DOI: 10.1109/MMAR.2015.7283901.

[14] A. Galuszka, T. Grzejszczak, J. Smieja, A. Olczyk, and J. Kocerka: On parallel conformant planning as an optimization problem. *32nd Annual European Simulation and Modelling Conference, Ghent*, (2018), 17–22.

[15] M. Ghallab *et al.*: PDDL – the Planning Domain Definition Language, Version 1.2. *Technical Report DCS TR-1165, Yale Center for Computational Vision and Control*, (1998).

[16] A. Grastien and E. Scala: Sampling Strategies for Conformant Planning. *Proc. Twenty-Eighth International Conference on Automated Planning and Scheduling*, (2018), 97–105.

[17] A. Grastien and E. Scala: CPCES: A planning framework to solve conformant planning problems through a counterexample guided refinement. *Artificial Intelligence*, **284** (2020), 103271, DOI: 10.1016/j.artint.2020.103271.

[18] D. Hoeller, G. Behnke, P. Bercher, S. Biundo, H. Fiorino, D. Pellier, and R. Alford: HDDL: An extension to PDDL for expressing hierarchical planning problems. *Proc. AAAI Conference on Artificial Intelligence*, **34**(6), (2020), 1–9, DOI: 10.1609/aaai.v34i06.6542.

[19] J. Koehler and K. Schuster: Elevator Control as a Planning Problem. *AIPS-2000*, (2000), 331–338.

[20] R. van der. Krogt: Modification strategies for SAT-based plan adaptation. *Archives of Control Sciences*, **18**(2), (2008).

[21] 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, DOI: 10.1016/j.camwa.2010.03.060.

[22] 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, DOI: 10.1109/MIS.2005.25.

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

[24] E.P.D. Pednault: ADL and the state-transition model of action. *Journal of Logic and Computation*, **4**(5), (1994), 467–512, DOI: 10.1093/logcom/4.5.467.

[25] 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, DOI: 10.1016/j.asoc.2010.10.014.

[26] F. Pommerening, G. Röger, M. Helmert, H. Cambazard, L.M. Rousseau, and D. Salvagnin: Lagrangian decomposition for classical planning. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, (2020), 4770—4774, DOI: 10.24963/ijcai.2020/663.

[27] T. Rosa, S. Jimenez, R. Fuentetaja, and D. Barrajo: Scaling up heuristic planning with relational decision trees. *Journal of Artificial Intelligence Research*, **40** (2011), 767–813, DOI: 10.1613/jair.3231.

[28] S.J. Russell and P. Norvig: *Artificial Intelligence: A Modern Approach*. Fourth Edition. Pearson, 2020.

[29] J. Seipp, T. Keller, and M. Helmert: Saturated post-hoc optimization for classical planning. *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, (2021).

[30] D.E. Smith and D.S. Weld: Conformant Graphplan. *Proc. 15th National Conf. on AI*, (1998).

[31] D.S. Weld: Recent Advantages in AI Planning. *AI Magazine*, (1999), DOI: 10.1609/aimag.v20i2.1459.

[32] D.S. Weld, C.R. Anderson, and D.E. Smith: Extending graphplan to handle uncertainty & sensing actions. *Proc. 15th National Conf. on AI*, (1998), 897–904.

[33] X. Zhang, A. Grastien, and E. Scala: Computing superior counter-examples for conformant planning. *Proc. AAAI Conference on Artificial Intelligence* **34**(6), (2020), 1–8, DOI: 10.1609/aaai.v34i06.6558.