Parallel Non-Deterministic Planning Research

Xinyan Liu
School of Computer, Guangdong University of Technology, Guangzhou 510006, China
411681211@qq.com

Abstract. This paper mainly introduces the theory of parallel Non-Deterministic planning domain, intelligent planning theory and a language RDDL describing the domain of parallel non-Deterministic planning. A first-order probability tree model is proposed for a specific domain, namely the semantic model of the elevator domain, which is helpful for solving problems in this domain.

Keywords: Non-Deterministic planning; RDDL; elevator domain.

1. Introduction

Planning in artificial intelligence (AI) research is one of its early research areas, dating back to the 1960s, Newell and Simon's Problem Solver (GPS) in 1957, Green's QA3 System[1], Fike in 1971 And Nilsson's STRIPS system[2] has an epoch-making significance in the field of intelligent planning, making planning very easy to describe and operate, but due to objective conditions, the field has been in a conservative state. In recent years, with the improvement of objective conditions, the field of intelligent planning has made tremendous progress.

The research content of intelligent planning is “to understand and analyze the surrounding environment, to implement reasoning on a number of alternative actions and resource constraints according to the objectives achieved by the plan, and to comprehensively formulate a reasonable action plan that can achieve the goal.” [3]. Parallel uncertain programming is one of the complex planning problems in the field of intelligent planning. Parallel probability planning can solve real-life problems in life. For example, the elevator field is one of the classic fields. This paper proposes a first-order probability tree semantic model to replace the traditional proposition to describe the domain problem.

2. Intelligent Planning

Planning problems can be broadly classified into classical planning problems and non-classical planning problems, and parallel non-Deterministic planning is a non-classical planning problem.

2.1 Classical Planning

The classical intelligent planning problem is based on strong constraints, requiring the initial state and action effects to be determined, the external world to be fully observable, and the action not to have duration. It is generally believed that the system model of the classical programming problem needs to satisfy the following eight assumptions:

1) Σ is limited, that is, the number of states contained in the state space that requires the problem is limited;
2) Σ is completely observable, that is, the agent has complete knowledge of each state; The state transition system of Σ is determined, that is the state transition of each state will result in a uniquely determined successor state;
3) The state transition system of Σ is determined, that is, the state transition of each state will result in a uniquely determined successor state;
4) Σ is a static system, that is, there is no event that changes the system. Only by performing actions can the state of the system be changed;
5) Σ specifies a restricted target, that is, the target condition can be specified to correspond to one or several states in the state space, and the action sequence reaching any one of the states

is the correct planning solution. Under this assumption, the state is Target constraints such as trajectory constraints and effect functions cannot be processed;
6) Linear programming solution, which requires the form of the planning solution to be a finite sequence of actions;
7) does not include a representation of time, that is, the occurrence of the required action and event does not contain duration, and the state transition is instantaneous;
8) Offline planning, that is, the system model is required to be stationary during the planning and solving process, and the planning solution is obtained according to the pre-specified initial state and target state, ignoring the changes occurring in the external world.

However, these constraints are too restrictive to real-world problems, hindering the application of intelligent planning in real life. In the real world, most of the above constraints are not established, so non-classical planning is required to expand.

2.2 Parallel Non-Deterministic Planning

2.2.1 Parallel Definition

Parallel planning can perform multiple actions [4] in one-time step. The advantages include: independent actions can be performed in parallel, regardless of the order between actions; the total time step of planning is greatly reduced. Parallel planning takes into account the mutual exclusion between actions. Whether due to conflict or demand competition, mutually exclusive actions cannot be performed at the same time, because this will make the world model unstable.

2.2.2 Non-Deterministic Planning

Non-Deterministic planning studies began in the 1970s[5]. The uncertainty includes the uncertainty of the initial state and the uncertainty of the action effect. A planning area can correspond to multiple specific planning issues. Each planning problem $\prod$ consisting of the planning domain description $\sum$ the initial state I and the target condition G, ie ($\prod =< \sum, \mathrm{I}, \mathrm{G} >$). The indeterminate action is an action with multiple sets of effects, and the multiple sets of effects have a common premise.

In non-Deterministic planning, the uncertainty of the action brings uncertainty in the system transformation. Probabilistic planning dominates in non-Deterministic planning, where the transformation is a transition probability distribution. In short, it has three characteristics:
1) The effect of the action is a set;
2) The evolution of the system after the action is a transfer distribution;
3) In the case of unobservable, the system is composed of multiple specific states in the state of faith.

3. Relational Dynamic Influence Diagram Language (RDDL)

RDDL is a rule-based programming description language whose semantics is Relational Dynamic Bayesian Network (RDBN) [6]. It has many new features that are not available in other programming languages. In the RDDL language, everything is a non-fluent or fluent argument with parameters, describing conditional probability functions, reward functions, objective functions, and states/actions through logical expressions or distribution functions. Constraint, etc. [7]. RDDL is based on the following principles:

- Everything is a parameterized variable (fluent or non-fluent)
  - Action fluents
  - State fluents
  - [Optional] Observation fluents (for partially observed domains)
  - Constant non-fluents (general constants, topology relations, . . .)
- Flexible fluent types
  - Binary (predicate) fluents
  - Multi-valued (enumerated) fluents

  - Integer and continuous fluents (numerical fluents from PDDL 2.1)
- The semantics is simply a ground Dynamic Bayes Net (DBN)
  - Supports factored state and observations
  - Supports factored actions, hence concurrency (and never conflicts!)
  - Supports intermediate state fluents for multi-layered DBNs
    - Express (stochastic) derived predicates (c.f., PDDL 1.2 and 2.2)
    - Express correlated effects
    - Stratification by levels enforces a well-defined relational multi-layer DBN
  - Naturally supports independent exogenous events
- General expressions in transition and reward functions
  - Logical expressions ($\wedge$, |, ~, =>, <=> plus $\exists/\forall$ quantification over variables)
  - Arithmetic expressions (+, -, $*$, / plus $\sum/\prod$ aggregation over variables)
  - (In)equality comparison expressions (==, ~=, <, >, <=, >=)
  - Conditional expressions (if-then-else, switch)
  - Basic probability distributions (Bernoulli, Discrete, Normal, Poisson, ...)
- Classical Planning as well as General (PO)MDP objectives
  - Arbitrary reward (goals, numerical preferences) (c.f., PDDL 3.0)
  - Finite horizon
  - Discounted or undiscounted
- State/action constraints
  - Encode legal actions (i.e., action preconditions)
  - Assert state invariants (e.g., a package cannot be in two locations)

4. Elevator Domain Semantic Model

The elevator domain is a typical description field for RDDL, and we provide first-order probability tree model to describe two states of this domain. Such a form is conducive to the solution of the problem.

4.1 Person-waiting-up

The person-waiting-up fluent is that, a person is waiting unless they get on an elevator going in their direction. The RDDL description is listed in Fig.1, and the FOPT is presented in Fig.2.

```
person-waiting-up'(?f) =
if (person-waiting-up(?f)
     ~exists_{?e: elevator} [elevator-at-floor(?e, ?f) ^ elevator-dir-up(?e) ^
~elevator-closed(?e)])
               then KronDelta(true)
               else Bernoulli(ARRIVE-PARAM(?f));
```

Fig. 1 RDDL for the person-waiting-up

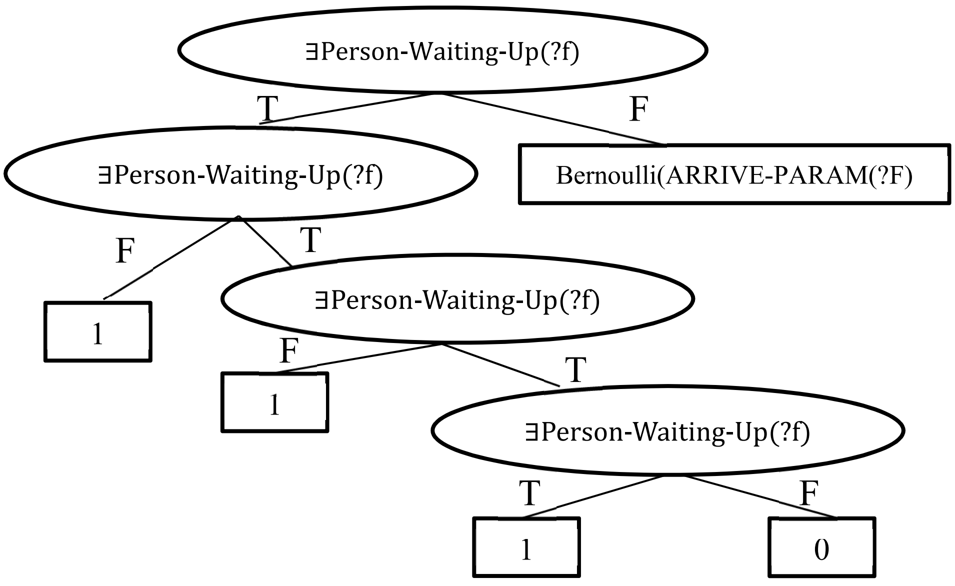

Fig. 2 FOPT for person-waiting-up

4.2 Elevator-at-floor

The elevator-at-floor fluent is that, which elevator at which floor is based on the movements of elevators. The RDDL description is listed in Fig.3, and the FOPT is presented in Fig.4

```
elevator-at-floor'(?e, ?f) =
if (~elevator-closed(?e) | ~move-current-dir(?e))
      then KronDelta( elevator-at-floor(?e, ?f) )
else if (move-current-dir(?e) ^ elevator-dir-up(?e) ^ exists_{?cur : floor}
       [elevator-at-floor(?e, ?cur) ^ ADJACENT-UP(?cur,?f)])
      then KronDelta(true)
      else
         if (move-current-dir(?e) ^ ~elevator-dir-up(?e) ^ exists_{?cur : floor}
              [elevator-at-floor(?e, ?cur) ^ ADJACENT-UP(?f,?cur)])
              then KronDelta(true)
         else if (move-current-dir(?e) ^ elevator-dir-up(?e)
                   ^ ~exists_{?next : floor}
              [elevator-at-floor(?e, ?f) ^ ADJACENT-UP(?f,?next)])
              then KronDelta( elevator-at-floor(?e, ?f) )
              else
                 if (move-current-dir(?e) ^ ~elevator-dir-up(?e)
                    ^ ~exists_{?next : floor}
                    [elevator-at-floor(?e, ?f) ^ ADJACENT-UP(?next,?f)])
                 then KronDelta( elevator-at-floor(?e, ?f) )
                 else  KronDelta( false );;
```

Fig. 3 RDDL for the elevator-at-floor

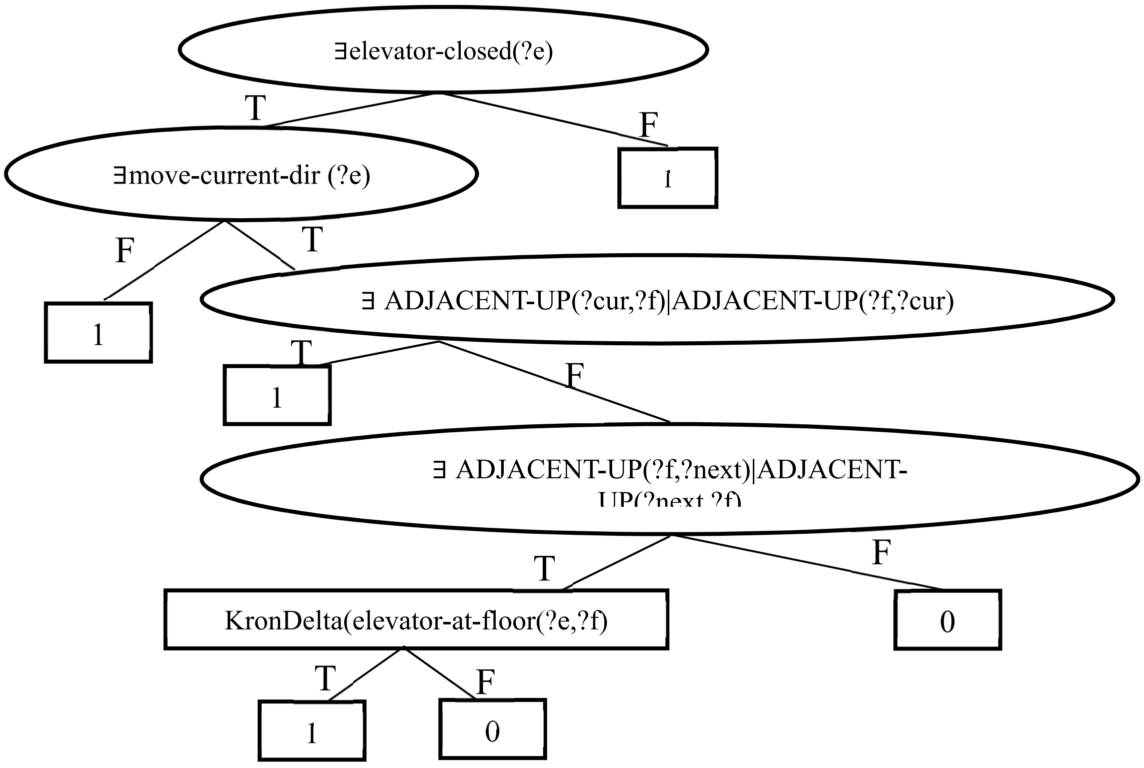
∃elevator-closed(?e)
T
F
∃move-current-dir (?e)
1
F
T
1
∃ ADJACENT-UP(?cur,?f)|ADJACENT-UP(?f,?cur)
T
F
1
∃ ADJACENT-UP(?f,?next)|ADJACENT-UP(?next,?f)
T
F
KronDelta(elevator-at-floor(?e,?f)
0
T
F
1
0

Fig. 4 FOPT for the elevator-at-floor

5. Summary

In this article, we introduce parallel non-determain planning. RDDL language, and give FOPT model in the elevator domain. I hope to give more general model in the future to further solve the parallel non-determain planning problem.

References

[1]. C Green. Readings in Artificial Intelligence. Elsevier Press,1981,p. 202-222.

[2]. Fikes R, Nilsson N. Strips: A new approach to the application of theorem proving to problem solving. IJCAI. Imperial College, London, England. September 1–3, 1971.p.189-208.

[3]. C Green: Theorem Proving by Resolution as a Basis for Question-Answering Systems (Machine intelligence, Stanford Research Institute, USA 1981). p183-205.

[4]. A Milani, M Terragnolo. Representing conflicts in parallel planning. Proceedings of the First Conference (AIPS 92). College Park, Maryland, June 15-17, 1992, p291-192.

[5]. A Cimatti, M Roveri, P Traverso. Strong Planning in Non-Deterministic Domains Via Model Checking. AAAI Press, 1998, p.36-43.

[6]. C Manfredotti, E Messina. Relational dynamic Bayesian networks to improve multi-target tracking. ACIVS 2009: Advanced Concepts for Intelligent Vision Systems. Bordeaux, France, 2009, p.528-536.

[7]. A Coles, A Coles, AG Olaya, S Jiménez, et al. A Survey of the Seventh International Planning Competition. AI Magazine. Vol. 33 (2012) No. 1, p.83-88.