Controlling Deliberation with the Success Probability in a Dynamic Environment

Seiji Yamada

Interdisciplinary Graduate School of Science and Engineering
Tokyo Institute of Technology
4259 Nagatsuda, Midori-ku, Yokohama, Kanagawa 226, JAPAN
E-mail: yamada@ymd.dis.titech.ac.jp

Abstract

This paper describes a novel method to interleave planning with execution in a dynamic environment. Though, in such planning, it is very important to control deliberation: to determine the timing for interleaving them, few research has been done. To cope with this problem, we propose a method to determine the interleave timing with the success probability, SP, that a plan will be successfully executed in an environment. We also developed a method to compute it efficiently with Bayesian networks and implemented $\mathcal{SIP}$ system. The system stops planning when the locally optimal plan's SP falls below an execution threshold, and executes the plan. Since SP depends on dynamics of an environment, a system does reactive behavior in a very dynamic environment, and becomes deliberative in a static one. We made experiments in Tileworld by changing dynamics and observation costs. As a result, we found the optimal threshold between reactivity and deliberation in some problem classes. Furthermore we found out the optimal threshold is robust against the change of dynamics and observation cost, and one of the classes in which $\mathcal{SIP}$ works well is that the dynamics itself changes.

Introduction

To select an action for an agent in a dynamic environment, reactive planning recently has become a significant topic in artificial intelligence and robotics (Agre & Chapman 1987)(Brooks 1986) (Georgeff & Lansky 1987). Some researchers argue no planning is necessary to intelligent behavior. However, we obviously require planning for more intelligent behavior including prediction, and need to integrate reactivity with deliberation. In this context, there is a significant issue: how to control deliberation in a dynamic environment. Unfortunately we have few promising solutions.

In this paper, we propose a novel method to interleave planning with execution in a dynamic environment. For controlling deliberation: determining the timing to switch planning to execution, a system uses the success probability, SP that it successfully executes a plan in an environment. A plan is represented with a Bayesian network, and we have developed a method to compute SP efficiently. Since SP depends on dynamics of an environment, our system does reactive behavior in a very dynamic environment, and becomes deliberative in a static one. Thus our approach integrates reactivity with deliberation depending on dynamics of an environment, and gives a solution to the above issue.

We implemented the $\mathcal{SIP}$ system for evaluating our approach, and made various experiments in the simplified Tileworld by changing dynamics and observation costs. As a result, we found the optimal threshold exists between reactivity and deliberation in some problem classes. Furthermore we found out the optimal threshold is independent of the change of dynamics and observation cost, and one of the classes in which $\mathcal{SIP}$ works well is that the dynamics itself changes.

Elegant studies have been done on learning an optimal policy with time constraints in stochastic automata, and the *envelop* was proposed to speedup the learning (Dean *et al.* 1993b)(Dean *et al.* 1993a). Furthermore the agent does *deliberation scheduling* for assigning the computational resource to making envelopes and optimizing a policy (Dean *et al.* 1993b)(Dean *et al.* 1993a). However no goal changes in an environment, and the time constraints are explicitly given to an agent. In a Tileworld where $\mathcal{SIP}$ is evaluated, goals constantly change and no explicit time constraint is given. Since the time constraints exist implicitly in an environment, an agent has to estimate them with observation.

Though McDermott first proposed the interleave planning (McDermott 1978), he did not describe any method to determine the timing of interleaving. Pollack made experiments in a Tileworld for investigating an IRMA model (Pollack & Ringuette 1990), and Kinny studied the relation between commitment and dynamics (Kinny & Georgeff 1991)(Kinny, Georgeff, & Hendler 1992). Unfortunately they did not deal with controlling deliberation. Boddy proposed the anytime algorithm which returns better answer as time passes (Boddy & Dean 1989). However they focused on a time restriction, and did not directly deal with environment dynamics. Subsumption architecture (Brooks 1986) tried to integrates reactivity with deliberation. Unfortunately it did not provide a general procedure to determine when a high-level process subsumed the low-level ones. Drummond's goal satisfaction probability (Drummond & Bresina 1990) is similar to ours. However the operators are too simple to represent complex causality.

Kirman investigated the prediction of real-time planner performance by domain characterization (Kirman 1994). Our work of characterizing the class in which $\mathcal{SIP}$ works well is concerned with his study. However, since his framework is defined on the Markov decision process and our domain including a Tileworld may not satisfy the Markov property, we consider that his approach is not straightforwardly applied to our domain.

Our study is also concerned with real-time search. RTA* (Korf 1990) is real time search which interleaves planning with execution. Though RTA* is constant-time planning, dynamics was ignored and no method to determine the interleave timing. DTA* (Russell & Wefald 1991) is a decision-theoretic search algorithm interleaving planning with actions. It control deliberation by estimating the possibility that the further search may overrule the current decision. In contrast with DTA*, $\mathcal{SIP}$ estimates the possibility that the current plan execution will be success. Furthermore DTA* dose not do modeling an environment. Ishida applied deliberation to improve the *Moving Target Search* (Ishida 1992). When an agent gets caught in the local minima in the utility function, it begins deliberation for escaping. The criterion for switching reactivity to deliberation is far different from $\mathcal{SIP}$, and no dealing with dynamics.

Interleaving planning with execution using SP

What is a criterion for determining the timing to switch planning to execution in a dynamic environment? In a very dynamic environment, we should switch them in short intervals. In contrast, the intervals should be longer in a more static environment. Thus we argue that "*When the success possibility of a plan execution keeps high, planning should be continued. The planning is stopped and the plan is executed when the possibility falls below a certain value*". We call the success probability SP, and a certain value an *execution threshold*. We developed a planning procedure based on the above claim, and call it $\mathcal{SIP}$ (Success probability-based Interleave Planning).

A domain

First we define a dynamic environment. The problem definition is generalized from real-time knowledge-based systems (Laffy *et al.* 1988) and the simplified Tileworld (Kinny & Georgeff 1991).

Definition 1 (a dynamic environment)
A dynamic environment where a $\mathcal{SIP}$ agent acts is a problem space where *goals* appear and disappear as time passes. Each goal G_i has *value* V_i and a $\mathcal{SIP}$ agent repeatedly tries to achieve a goal before it disappears and obtains the value. The *agent's purpose* is to get as high total value as possible. □

Next we define operators and plans used in $\mathcal{SIP}$. Note that $\mathcal{SIP}$ uses a single operator, called a *goal operator*, for planning. In the followings, $+P$ and $-P$ mean a true and a negation value of a propositional variable P. $\neg P$ means negation of P.

Definition 2 (a goal-operator and plans) *The goal-operator* O achieves a goal and gets the value. It is a STRIPS-like operator consisting of a cond-list C, a delete-list D and an add-list A. A_i of O_i includes a *success literal* s_i which means obtaining the value of O_i's goal. The operator also has an *execution-time function* $et(O)$ which returns the time taken for executing it. A *plan* is a sequence of instantiated goal-operators, $[O_1, \ldots, O_n]$ describing order of goals to be achieved. A literal L in C_i, D_i, A_i of O_i are characterized with L_{C_i}, L_{D_i}, L_{A_i}. They are called a *cond-literal*, a *delete-literal* and an *add-literal* . □

Concrete methods for executing a goal-operator like path-planning are described depending on a domain and given as input. In $\mathcal{SIP}$, *deliberation* means planning with goal-operators: scheduling an optimal goal order to be achieve. *Reactivity* means reflective action of an agent without such scheduling.

$\mathcal{SIP}$: Interleave planning with SP

A $\mathcal{SIP}$ agent consists of an observer, an environment modeler, a $\mathcal{SIP}$ planner and a plan executor as shown in Fig.1. An observer constantly obtains data from an environment parallel to other modules, and gives the state descriptions (literals) to an environment modeler and a $\mathcal{SIP}$planner. Using them, an environment modeler estimates the persistence probabilities, and gives them to $\mathcal{SIP}$ planner. The $\mathcal{SIP}$ planner obtains state descriptions from an observer as an initial state, and generates a plan.

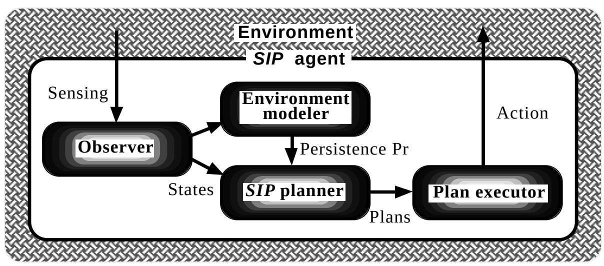

Figure 1 A $\mathcal{SIP}$ agent

The detailed $\mathcal{SIP}$ procedure is shown in Fig.2. The basic strategy of planning is forward beam search with an evaluation function: *expected value*. First a $\mathcal{SIP}$ system obtains initial states from an observer (*Observation* in Fig.2), and gets the persistence probabilities from an environment modeler (*Getting environment structure* in Fig.2). Next the planner applies goal-operators to the current state, and expands all new states. At every depth of planning, the planner computes SP and the expected values of all the expanded plans, and selects w plans with the highest expected values, where w is width of beam search.

```
A procedure SIP(G, w, τ )
  (G: a goal state,w : a beam width, τ : an execution threshold)
  while true do
    begin
      Observation;
      Getting environment structure;
      CS ← {(the observed state, [ ])};
      P ← [ ];            % CS is a set of sp-pair: (State, Plan).
      MSP ← 2;
      while MSP > τ ∧ P is not a complete plan in CS do
        begin
          NS ← all sp-pairs expanded from CS
                  with operator applications;
          Computing SP and expected values for NS;
          CS ← w sp-pairs with high expected values in NS;
          MSP ← the success probability of the plan P
                  with the maximum expected value in CS
        end
      [O1, ..., On ] ← P ;
      i ← 1;
      while i ≠ n +1 ∧ the literal s of Oi −1 is achieved do
        begin
          executing Oi ;
          i ← i +1
        end
    end
```

Figure 2 A $\mathcal{SIP}$ procedure

If SP of an (locally) optimal plan which has the highest expected value falls below an execution threshold or any complete plan is generated, $\mathcal{SIP}$ will stop planning and execute the optimal plan. If not, the selected w states will be expanded, and planning will start again.

The plan executor executes operators in order of a plan until any execution fails or all of them are successfully executed. The above cycle is repeated. The *complete plan* includes all of the observed goals, and the *partial plan* is not complete one. Forward chaining guarantees that any partial plan is executable in an initial state, and beam search reduces a search space. w and an execution threshold are given to a system as input. This procedure realizes our claim.

Planning depth is controlled by changing an execution threshold $\tau \in [0, 1]$. When τ is high, planning depth becomes short and the behavior becomes reactive, whereas when it is low, the planning depth is long and the behavior becomes deliberative. For example, Fig.3 shows the behavior of the optimal plan's SP as the plan grows. When an execution threshold is 0.8, the plan is executed at 2 steps, and if it is 0.3, the execution is done at 6 steps.

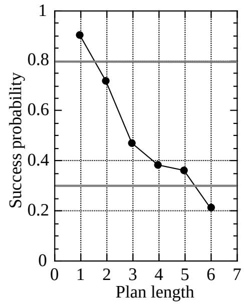

Figure 3 SP and an execution threshold

The success probability of a plan

We define the success of an operator execution, a plan execution and an expected value. In the followings, $Pr(A, B)$ means $Pr(A \wedge B)$.

Definition 3 (success of operator execution) An execution of a goal-operator O_i is success *iff* the O_i's success literal s_i becomes true in an environment after the execution. A proposition S_i means success of O_i's execution. $\square$

Definition 4 (success probability) A plan execution is success *iff* all executions of operators in a plan are success: $S_1 \wedge \cdots \wedge S_n$ becomes true after a plan $[O_1, \ldots, O_n]$ is executed. $Pr(+S_1, \ldots, +S_n)$ is a *success probability* SP of a plan. $\square$

Note that Def. 4 is available for both a complete plan and a partial plan. With an execution procedure in Fig. 2, we define an expected value of a plan.

Definition 5 (expected value of a plan) An *expected value* $E[V]$ of a plan $[O_1, \ldots, O_n]$ is $Pr(+S_1) \cdot V_1 + Pr(+S_1, +S_2) \cdot V_2 + \cdots + Pr(+S_1, \ldots, +S_n) \cdot V_n$, where V_i is the value of O_i's goal. $\square$

The plan Bayesian networks

For representing probabilistic causality between events and computing the success probability, we introduce the *plan Bayesian network*.

Definition 6 (temporal proposition) $\langle L, t \rangle$ is a *temporal proposition* that means a literal L is true at a time t in an environment. $\square$

Definition 7 (causal relation and time points) If L_{C_j} is L_{A_i} added by O_i in a plan, there will be a *causal relation* $L_{A_i} \prec L_{C_j}$. For plan $[O_1, \ldots, O_n]$, t_0 is an *execution start time* of O_1, $t_i = t_0 + \sum_{k=1}^{i} et(O_k)$ is an *execution finish time* of O_i, and $t(L)$ is a function returning the time when the observer observed that a literal L became true in an environment. $\square$

$\mathcal{SIP}$ needs the following input probabilities.

Definition 8 (input probabilities)

- *Effect probability,* $E\text{-}Pr(O_i, L)$: A probability that an O_i's add-literal L becomes true in an environment after executing O_i. This means the certainty of the operator's effect.
- *Observation probability,* $O\text{-}Pr(L)$: A probability that an observed literal L was really true in an environment. This means the certainty of information obtained by the observer.
- *Persistence probability,* $P\text{-}Pr(L, T)$: A probability that a literal L is yet true when the time T has passed from when it became true in an environment. This means the degree of the change in an environment. □

Using a plan, time points and above input probabilities, we completely construct the plan Bayesian network, PBN. PBN is described with a Bayesian network (Pearl 1988) widely used for representing probabilistic causality.

Next we explain how to construct the PBN. In the followings, $\boldsymbol{V}$, $\boldsymbol{E}$ are sets of nodes and edges, and $e(v_1, v_2) \in \boldsymbol{E}$ stands for a directed edge $v_1 \rightarrow v_2$. $BEL(x)$ is a vector $(Pr(+x|\boldsymbol{e}), Pr(-x|\boldsymbol{e}))$ ($\boldsymbol{e}$ is conjunction of evidences), and a conditional probability assigned to $e(x, y)$ is $M_{y|x} = \begin{pmatrix} Pr(+y|+x) & Pr(-y|+x) \\ Pr(+y|-x) & Pr(-y|-x) \end{pmatrix}$. A proposition $Ob(L, t)$ means an observation that a literal L became true at a time point t, and $Ex(O)$ means that an goal-operator O is executable in an environment. The time point t_i and $t(L)$ were described in Def. 7.

Definition 9 (plan Bayesian network) The *plan Bayesian network*, PBN, of a plan $[O_1, \ldots, O_n]$ is a directed acyclic graph consisting of $\boldsymbol{V}$, $\boldsymbol{E}$ and $M_{y|x}$ in the followings, where $1 \leq i \leq n$.

- $\boldsymbol{V}$ and $\boldsymbol{E}$:

  (*execution-node*): $\langle Ex(O_i), t_{i-1}\rangle \in \boldsymbol{V}$.

  (*cond-node*): $\langle L_{C_i}, t_{i-1}\rangle \in \boldsymbol{V}$, $e(\langle L_{C_i}, t_{i-1}\rangle, \langle Ex(O_i), t_{i-1}\rangle) \in \boldsymbol{E}$.

  (*add-node*): $\langle L_{A_i}, t_i\rangle \in \boldsymbol{V}$, $e(\langle Ex(O_i), t_{i-1}\rangle, \langle L_{A_i}, t_i\rangle) \in \boldsymbol{E}$.

  (*delete-node*): $\langle \neg L_{D_i}, t_i\rangle \in \boldsymbol{V}$, $e(\langle Ex(O_i), t_{i-1}\rangle, \langle \neg L_{D_i}, t_i\rangle) \in \boldsymbol{E}$.

  (*observation-node*): If $\langle L, t_{i-1}\rangle$ was observed, then $\langle Ob(L), t(L)\rangle \in \boldsymbol{V}$ and $e(\langle Ob(L), t(L)\rangle, \langle L, t_{i-1}\rangle)$.

  (*relation-edge*): If $L_{A_i} \prec L_{C_j}$ exists, then $e(\langle L_{A_i}, t_i\rangle, \langle L_{C_j}, t_{j-1}\rangle) \in \boldsymbol{E}$. If $L_{D_i} \prec \neg L_{C_j}$ exists, then $e(\langle \neg L_{D_i}, t_i\rangle, \langle \neg L_{C_j}, t_{j-1}\rangle) \in \boldsymbol{E}$.

- *Conditional probabilities*:

  (*observation-pr*): $BEL(\langle Ob(L),\ t(L)\rangle) = (O\text{-}Pr(L), 1 - O\text{-}Pr(L))$.

  (*effect-pr*): If $x = \langle Ex(O_i), t_{i-1}\rangle$ and y is the child node of x, then
  $M_{y|x} = \begin{pmatrix} E\text{-}Pr(O_i, L) & 1 - E\text{-}Pr(O_i, L) \\ 0 & 1 \end{pmatrix}$.

  (*persistence-pr*): If $x = \langle L_{A_i}, T_1\rangle$ or $\langle Ob(L), T_1\rangle$, and $y = \langle L_{C_j}, T_2\rangle$, then
  $M_{y|x} = \begin{pmatrix} P\text{-}Pr(L, T_2 - T_1) & 1 - P\text{-}Pr(L, T_2 - T_1) \\ 0 & 1 \end{pmatrix}$.

  (*cond-pr*): If an execution-node has cond-nodes $c_i, \ldots, c_m$, then
  $Pr(+Ex|c_1, \ldots, c_m) = \begin{cases} 1 & \text{if } +c_1 \wedge \cdots \wedge c_m \\ 0 & \text{otherwise} \end{cases}$,
  $Pr(-Ex|c_1, \ldots, c_m) = \begin{cases} 0 & \text{if } +c_1 \wedge \cdots \wedge c_m \\ 1 & \text{otherwise} \end{cases}$. □

We use two assumptions for the above definitions.

A1: Cond-nodes and add-nodes of the same operator are mutually probabilistic independent.

A2: Execution-node is true *iff* all of its cond-nodes are true.

Though the assumptions may slightly restrict representation power of PBN, they make the computation of SP very efficient as mentioned in next section. Fig.4 shows a plan Bayesian network constructed from a plan $P = [O_1,\ O_2,\ O_3] = [([a_{C_1}, b_{C_1}], [c_{D_1}], [d_{A_1}]),\ ([d_{C_2}, \neg e_{C_2}],\ [], [f_{A_2}]),\ ([d_{C_3}, f_{C_3}, \neg c_{C_3}], [], [g_{A_3}])]$. The causal relation is $\{ d_{A_1} \prec d_{C_3},\ c_{D_1} \prec \neg c_{C_3},\ f_{A_2} \prec f_{C_3} \}$.

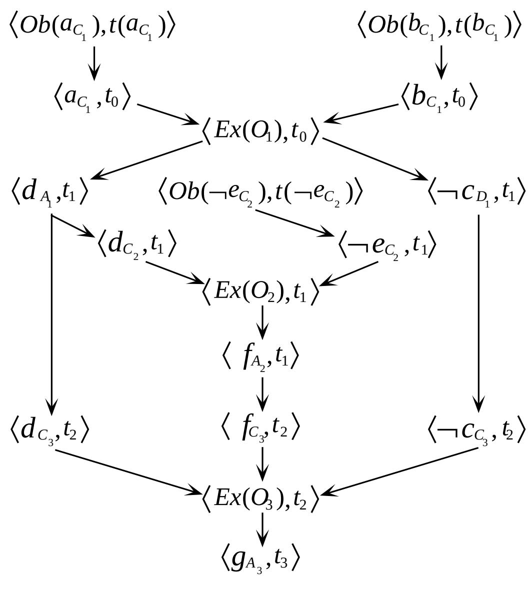

Figure 4 The plan Bayesian network

Computing the success probability

Using a temporal proposition, the success of operator execution, S_i in Def. 4, is described as $\langle s_i, t_i \rangle$. Thus, with Def. 3 and Def. 4, we describe SP of executing a plan $P = [O_1, \ldots, O_n]$ at time point t_0 as $SP(P, t_0) = Pr(+\langle s_1, t_1 \rangle, \ldots, +\langle s_n, t_n \rangle)$. The t_0 is the start time point to execute a plan.

$SP(P, t_0)$ is expanded in the followings. Equation (1) is obtained with a chain rule, where $\boldsymbol{e}_i = +\langle s_1, t_1 \rangle \wedge \cdots \wedge +\langle s_i, t_i \rangle$. Equation (2) and (3) are obtained from a method developed in (Pearl 1988) under A1 and A2. Describing the essence of the derivation, as computing $Pr(+\langle s_i, t_i \rangle \mid +\langle s_1, t_1 \rangle, \ldots, +\langle s_{i-1}, t_{i-1} \rangle)$, the condition events $+\langle s_1, t_1 \rangle, \ldots, +\langle s_{i-1}, t_{i-1} \rangle$ block all of the relevant loops and make the PBN singly-connected. Consequently we can straightforward apply an efficient and exact method (Pearl 1988) to compute SP.

$$\begin{aligned} SP(P, t_0) &= Pr(+\langle s_1, t_1 \rangle, \ldots, +\langle s_n, t_n \rangle) \\ &= \prod_{i=1}^{n} Pr(+\langle s_i, t_i \rangle | \boldsymbol{e}_{i-1}) \qquad (1) \end{aligned}$$

$$\begin{aligned} &Pr(+\langle s_i, t_i \rangle | \boldsymbol{e}_{i-1}) \\ &= E\text{-}Pr(O_i, +\langle s_i, t_i \rangle) \prod_{L \in C_i} Pr(+\langle L_{C_i}, t_{i-1} \rangle | \boldsymbol{e}_{i-1}) \qquad (2) \end{aligned}$$

$$\begin{aligned} &Pr(+\langle L_{C_i}, t_{i-1} \rangle | \boldsymbol{e}_{i-1}) \\ &= \begin{cases} P\text{-}Pr(L, t_{i-1} - t_h) & \text{(a)} \\ E\text{-}Pr(O_h, L) \cdot P\text{-}Pr(L, t_{i-1} - t_h) & \text{(b)} \\ O\text{-}Pr(L) \cdot P\text{-}Pr(L, t_{i-1} - t_h) & \text{(c)} \end{cases} \qquad (3) \end{aligned}$$

(a) $\sim$ (c) in equation (3) are in the followings, where N is a parent node of $\langle L_{C_i}, t_{i-1} \rangle$ and t_h is the time point.

(a): N is a node of a literal s of $\langle Ex(O_1), t_0 \rangle \sim \langle Ex(O_{i-1}), t_{i-2} \rangle$ or any cond-node of $\langle Ex(O_1), t_0 \rangle \sim \langle Ex(O_{i-1}), t_{i-2} \rangle$ is a brother node of $\langle L_{C_i}, t_{i-1} \rangle$.

(b): Not (a), and N is an add-node or a delete-node of $Ex(O_h, t_h)$.

(c): Not (a), and N is an observation-node.

With equation (1), (2), and (3), SP is computed incrementally as growing plans, and the time complexity for one step of a plan is *constant*. Thus, the time complexity to compute SP of a n step plan is $O(n)$. In contrast that the complexity for updating probabilities in Bayesian networks is generally *NP-hard* (Cooper 1990), computing SP on PBN is very efficient. Furthermore, since input probabilities depend on dynamics and an observation cost, $\mathcal{SIP}$ is able to deal with them.

Experiments in the Tileworld

We made experiments in the simplified Tileworld (Kinny & Georgeff 1991), a standard test-bed for a dynamic environment (Pollack & Ringuette 1990). The simplified Tileworld is a chess-board-like grid in which there are agents, obstacles and holes (see Fig.5). An agent can move up, down, left or right, one cell at a time. An obstacle is a immovable cell. A hole (a goal) is a cell with a score (a value). By moving to the hole, an agent obtains the score, and then the hole disappears. Holes appears or disappear as time passes.

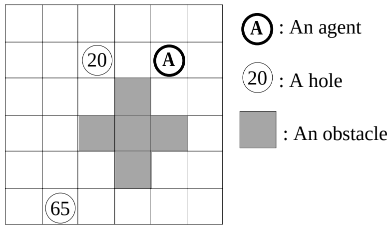

Figure 5 A simplified Tileworld

One of the purposes of the experiments is to find out the class of problems in which $\mathcal{SIP}$ outperforms both a reactive system and a deliberative one. In other words, it is to find out the problems in which the optimal execution threshold exists neither near to 0 nor 1.

The another purpose is to characterize the classes in which $\mathcal{SIP}$works well. Since $\mathcal{SIP}$ estimates the degree of the dynamics on-line, it is adaptive to the change of dynamics. Thus we attempt to characterize the class in which $\mathcal{SIP}$ outperforms the interleave planning which is not adaptive.

There is few experiments for examining the trade-off between deliberation and reactivity, and the adaptation of planning to the change of dynamics. Thus the experimental results are significant for designing an agent in a dynamic domain.

Parameters and procedure

We characterize the simplified Tileworld using the following properties: (1) Dynamics of an environment, (2) Uncertainty of actions including observations, (3) Costs for planning and observations. We selected necessary parameters for the properties and gave the default setting.

Parameters to be examined:

- *Dynamics*, $d = 1, 2, \ldots, 8$: The rate at which the world changes for an agent.
- *observation cost*, c: The time taken for an observation.
- *Execution threshold*: $\tau = 0, 0.1, \ldots, 1$

Agent parameters:

- *Execution time for a single moving*: 2 (fixed)
- *A goal-operators*: This describes a move from a hole to other holes. For path planning, an agent uses hill-climbing with Manhattan distance as an evaluation function.

- *Width of beam search*, $w = 4$. We fixed it since no significant change was observed by changing it.
- *Input probabilities*: Observation probability is $1 - u$, and effect probability that an agent moves successfully for p Manhattan distance is $(1 - u)^p$. We use a simple persistence probability function, $P\text{-}Pr(L,t) = \begin{cases} 1 & (t < b) \\ rt - rb + 1 & (b \le t < b + \frac{1}{r}) \\ 0 & (b + \frac{1}{r} \le t) \end{cases}$ derived from hole life-expectancy[1].
- *Uncertainty*, $u = 0.01$: The probability u that one-step moving and an observation fails. Failure of moving means no moving. If an observation fails, the object is observed randomly in one of the four neighbor (up, down, left and right) cells on the true position.

Environment parameters:

- *Grid size*: 20×20. No obstacle.
- *Hole scores*: Chosen from a uniform distribution for [60, 100].
- *Hole life-expectancy*: The interval between an appearance and a natural disappearance of a hole. Chosen from a uniform distribution for [1200, 5200].
- *Hole gestation time*: The interval between the appearances of successive holes. Chosen from a uniform distribution [100, 300].
- *Initial holes' positions*: The initial holes positions are randomly set.

Since the complexity for computing SP for one step is constant, we defined a unit of agent-time as the time for expanding a node in planning. The c time-units are taken by one observation, and d environment-time passes as one agent-time passes. For simplicity, we assumed an agent knows true distance to a target hole, thus no obstacle was necessary. An agent actually moves using hill-climbing. We used the *scoring rate* $\varepsilon = \frac{\textit{obtained score}}{\textit{the maximum possible score it could have achieved}}$ as performance measurement. Through all experiments, a terminal condition was that the standard deviation of ε converges to 0.01, and five results of the identical parameters were averaged. In the following, problems are described with the difference from default setting.

Exp-A: Changing dynamics and an observation cost

We investigated influence of dynamics and an observation cost on an agent's performance, and attempted to find out the optimal execution threshold was between 0 and 1: the most deliberative and the most reactive. For simplicity, we gave persistence probabilities to a $\mathcal{SIP}$ agent through these experiment. The environment modeler of $\mathcal{SIP}$ will be implemented in the next section.

[1] $[b, b + \frac{1}{r}]$ is equal to the range of hole life-expectancy, [1200, 5200], mentioned later.

Fist we changed an observation cost c for 5, 50, 100 and 200. Due to space constraints, the typical experimental result ($c = 100$) is shown in Fig.6. The results for $c = 5$, 50, 200 were almost similar to Fig.6. The x-axis and y-axis stand for an execution threshold τ and a scoring rate ε respectively. Through four observation costs, it is natural that a scoring rate decreases as dynamic increases. In Fig.6, when an environment is static ($d = 1$), the scoring rates are high independent of an execution threshold. However, for $d = 2$, 3 and 4, the scoring rates near to 0 and 1 decrease. Consequently we found out that the optimal execution threshold existed between reactivity and deliberation in some problems. Note that there is a *single peak* of a scoring rate in most of the problems. This is important for a hill-climbing method to search the optimal execution threshold. We observed these properties also in other results.

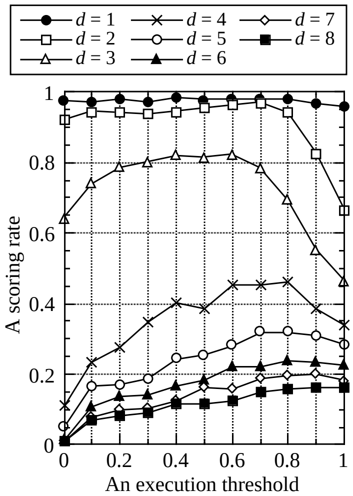

Figure 6 Changing dynamics ($c = 100$)

For $c = 50$, 100 and 200, there are the peaks between 0 and 1, and the rank correlation coefficients of any different dynamics pair in the same observation cost are positive, 0.2$\sim$0.5. Hence we consider the optimal execution thresholds are robust against the change of dynamics. Furthermore, since the peaks are around 0.7 in Fig.7 showing scoring rates averaged over dynamics, it is robust also against the change of an observation cost. These properties derives from SP is computed depending on dynamics and an observation cost.

The environment modeler

We implemented the environment modeler for advanced experiments. In $\mathcal{SIP}$, the environment modeling means the estimation of persistence probabilities with data from the observer. The data include the observed life-expectancies. The persistence probabilities

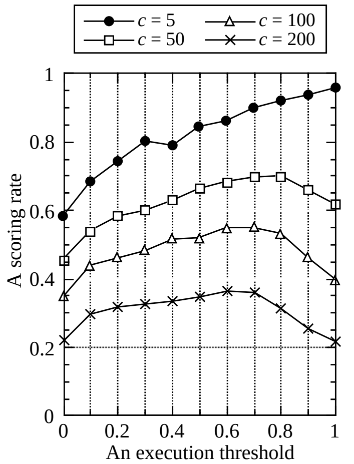

Figure 7 Averaged scoring rates to dynamics

significantly depend on dynamics of an environment. A following equations (Dean & Kanazawa 1989) were used for estimating d and r of $P\text{-}Pr(L,t)$ described earlier, and the modeling can be constantly done parallel to planning, execution and observation. The input is U: a set of last n samples of observed life-expectancies. The modeler is able to update the estimated value because the sample is constantly updated. We assume the agent knows the model of persistence probabilities and can use a parametric method. If the assumption is not hold, the error of estimation will increase.

$$\begin{aligned} d(U) &= \text{the minimum of } U \\ r(U) &= \frac{-\ 0.5}{\text{the average of } U - d(U)} \end{aligned}$$

Exp-B: Adaptation to the change of dynamics

$\mathcal{SIP}$ is adaptive to the change of dynamics because of the environment modeler. Hence we were interested in comparing $\mathcal{SIP}$ with fixed-depth planning which stops the planning at the given depth. Though the fixed-depth planning can control deliberation, it is not adaptive to the change of dynamics because of fixed depth. The same parameters of Exp-A were used, except changing the fixed-depth from 1 to 10 instead of an execution threshold. This fixed-depth planner is same to $\mathcal{SIP}$, except that the planning depth is fixed.

The results for $c = 100$ is shown in Fig.8. Comparing with Fig.6, the maximum scoring rates are almost equal to ones in Fig.6. This is showed also in other c. Unlike the fixed depth planning, $\mathcal{SIP}$ can change the plan length on-line even with a fixed execution threshold. Unfortunately, the advantage is not shown[2].

[2]We consider since the causality between goals is too simple, SP is smoothly decreased and the interleave timing of two planning are not much different.

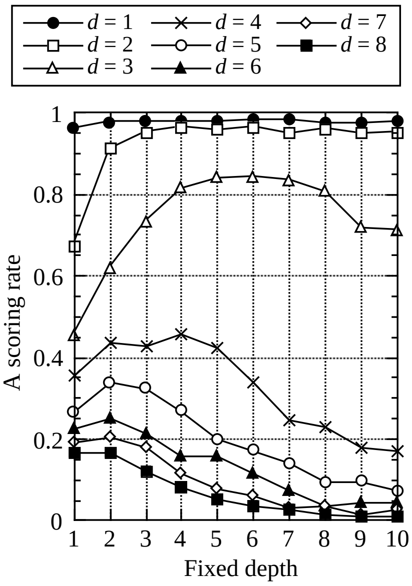

Figure 8 Fixed-depth planning

We expected that the optimal threshold of $\mathcal{SIP}$ would be more robust than the optimal depth of fixed-depth planning. It is because fixed-depth planning does not deal with dynamics, and it may change the optimal depth widely depending on the change of dynamics. Fig.9 shows the differences between ε at $\tau = 0.6$ or $depth = 6$ (which are the optimal conditions at $d = 3$) and the maximum ε, at $d = 3$, 4, 5 and 6. Smaller the difference is, more robust the system is. Thus we see $\mathcal{SIP}$ is more robust than fixed-depth planning.

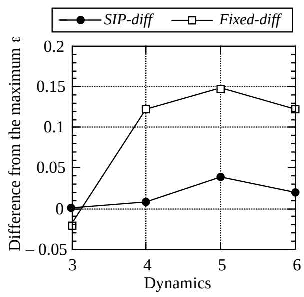

Figure 9 Robustness of $\mathcal{SIP}$

Next, we investigated the adaptation of $\mathcal{SIP}$ to the change of dynamics. Using $\mathcal{SIP}$ with the environment modeler and fixed-depth planner, we made the experiment in which dynamics itself changed. The dynamics was initially three, changed to four at 10000 agent-time, and to five at 20000 agent-time. The sample number n of a modeler was set 20 and $c = 100$. The results are shown in Fig.10. Though the fixed-depth planner outperformed $\mathcal{SIP}$ until 10000 time, after dynamics changed twice, $\mathcal{SIP}$ is better than the fixed-depth because of its adaptation to the change of dynamics. Thus we consider one of the classes in which $\mathcal{SIP}$ works well is the environment where dynamics itself changes.

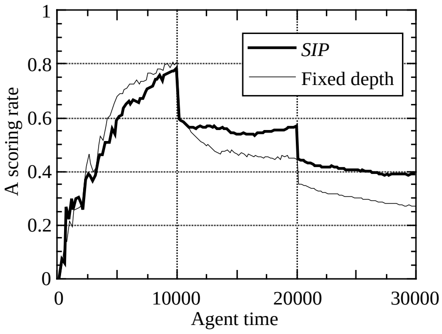

Figure 10 Adaptation to the change of dynamics

Conclusion

We proposed a $\mathcal{SIP}$ method for controlling deliberation in a dynamic environment. $\mathcal{SIP}$ computes the success probability SP that a plan will be executed successfully, and stops planning when the SP falls below an execution threshold. A plan is transformed into a plan Bayesian network, and the SP is efficiently computed on it. We made experiments in the simplified Tileworld for evaluating $\mathcal{SIP}$. As a result, we found out $\mathcal{SIP}$ worked better than a reactive and a deliberative system in some problems. Furthermore we found out the optimal threshold is robust, and one of good classes for $\mathcal{SIP}$ is where dynamics itself changes. However $\mathcal{SIP}$ needs to deal with replanning, and we need systematic characterization, like (Kirman 1994), of the classes where $\mathcal{SIP}$ works well.

Acknowledgments

Many thanks to Yoshinori Isoda at NTT Human Interface Laboratories for useful discussion.

References

Agre, P. E., and Chapman, D. 1987. Pengi: A implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 268–272.

Boddy, M., and Dean, T. 1989. Solving time-dependent planning problems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 979–984.

Brooks, R. A. 1986. A robust layered control system for a mobile robot. *IEEE Transaction on Robotics and Automation* 2(1):14–23.

Cooper, G. F. 1990. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence* 42:393–405.

Dean, T., and Kanazawa, K. 1989. Persistence and probabilistic projection. *IEEE Transaction on Systems, Man, and Cybernetics* 19(3):374–385.

Dean, T.; Kaelbling, J. K.; Kirman, J.; and Nicholson, A. 1993a. Deliberation scheduling for time-critical sequential decision making. In *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, 309–316.

Dean, T.; Kaelbling, J. K.; Kirman, J.; and Nicholson, A. 1993b. Planning with deadlines in stochastic domains. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 574–579.

Drummond, M., and Bresina, J. 1990. Anytime synthetic projection: Maximizing the probability of goal satisfaction. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 138–144.

Georgeff, M. P., and Lansky, A. L. 1987. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 677–682.

Ishida, T. 1992. Moving target search with intelligence. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 525–532.

Kinny, D., and Georgeff, M. 1991. Commitment and effectiveness of situated agents. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, 82–88.

Kinny, D.; Georgeff, M.; and Hendler, J. 1992. Experiments in optimal sensing for situated agent. In *Proceedings of the Second Pacific Rim International Conference on Artificial Intelligence*, 1176–1182.

Kirman, J. 1994. *Predicting Real-Time Planner Performance by Domain Characterization.* Ph.D. Dissertation, Brown University.

Korf, R. E. 1990. Real-time heuristic search. *Artificial Intelligence* 42:189–211.

Laffy, T. J.; Cox, P. A.; Schmidt, J. L.; Kao, S. M.; and Read, J. Y. 1988. Real-time knowledge-based systems. In *AI magazine*, volume 9. 27–45.

McDermott, D. 1978. Planning and action. *Cognitive Science* 2:77–110.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann.

Pollack, M. E., and Ringuette, M. 1990. Introducing the tileworld: Experimentally evaluating agent architectures. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 183–189.

Russell, S., and Wefald, E. 1991. *Do the Right Thing.* MIT Press.