Knowledge-Based Control of Decision-Theoretic Planning – Adaptive Planning Model Selection –

Jun Miura and **Yoshiaki Shirai**
Dept. of Computer-Controlled Mechanical Systems, Osaka University
Suita, Osaka 565-0871, Japan
email: jun@mech.eng.osaka-u.ac.jp
URL: http://www-cv.mech.eng.osaka-u.ac.jp/~jun

Abstract.

This paper proposes a new planning architecture for agents operating in uncertain and dynamic environments. Decision-theoretic planning has been recognized as a useful tool for reasoning under uncertainty; it calculates an optimal plan using a given *planning model* (state set, action set, probability distributions over possible state transitions, and utility function). In a dynamic environment, however, the current situation may be different from what an agent expects and the current planning model may not be feasible. It is, therefore, important for an agent to continuously examine the situation and to use an appropriate planning model. For this purpose, we propose to employ a knowledge-based meta-level reasoner to on-line select an appropriate planning model for an object-level decision-theoretic planning, based on the given knowledge of classification of the situation. This architecture could also be effective in reducing the computational cost of decision-theoretic planning by limiting the search space according to the situation. Two applications of the architecture to a dynamic robot planning and to a decision-making in highway driving show the generality and the usefulness of the architecture.

1 Introduction

To design planning algorithms for an agent that operates in the real world, we need to consider the following two issues: uncertainty and limitation of computational resources. Various activities of an agent such as sensing and motion inherently include uncertainty; an agent's computational power is definitely limited. It is, therefore, important for an agent to cope with uncertainties without largely increasing the computational cost.

Decision-theoretic planning [2] has been recognized as a useful tool for reasoning under uncertainty; it is usually defined by the following:

- state set $\mathcal{S}$.
- action set $\mathcal{A}$.
- probability distribution over the possible state transitions for executing action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. Exogenous changes are also included here, if any.
- utility function to evaluate each state or each state-action pair.

In this paper, we collectively call them a *planning model*. Under this model, an agent usually determine an action (or action sequence) which maximizes the expected utility.

Many works have adopted decision-theoretic planning to planning tasks under uncertainty (e.g., [11] [6] [13]). These works, however, assume that the environment is static.

Decision-theoretic planning is usually costly because it has to consider multiple outcomes of actions and the planning cost often increases exponentially to the search depth. This is a drawback when used in a dynamic environment where the allocated time for planning is limited. Recently Markov decision processes (MDPs) have been attracting much interests as a basic representation for planning under uncertainty [7]. Although several approaches (e.g., [4]) have been developed to efficiently obtain optimal policies for MDP problems, they still seem inappropriate for large-sized planning problems under time pressure.

One approach to reducing the computational cost is to properly control the allocation of computing resources to each decision-theoretic planning activity. Many works have recently been focusing on the concept of *limited rationality* [15], in which the cost of planning is explicitly considered and computational resources for object-level planning is allocated so that the overall utility including both plan efficiency and planning cost is maximized. Some of examples are: flexible computation [10], decision-theoretic meta-level control of (object-level) reasoning [15], and expectation-driven iterative refinement (EDIR) using anytime algorithms [3] [14].

These works are mainly interested in optimal allocation of computing resources *within a given planning model*. In a dynamic environment, however, we have to cope with the change of situation by switching planning models. Since a planning using a wrong model may lead to a fatal situation, it is important to frequently examine if the current model is fit to the current situation and to switch to an appropriate model, whenever necessary.

It could be possible to have a very large model which covers all possible situations. However this approach may not desirable due to the following two reasons. First, adopting a large model is computationally expensive in both model generation and model utilization because the number of transition relationships between states grows exponentially to that of states. Second, considering all possibility at once could sometimes hide the underlying structure of a planning problem.

Figure 1. Examining the whole problem space or examining only a part of the space.

An example of such a case will be shown in the next section.

Based on the above discussion, we propose to put a knowledge-based model selector on top of an ordinary decision-theoretic object-level planner. Given the knowledge of the structure of the planning problem, the model selector selects an appropriate planning model, thereby directing the efforts only to a limited, effective (or correct) computation (see Figure 1).

Many layered architectures have been proposed for controlling autonomous robots. Gat [9] proposed a three-level control architecture. In his architecture, called ATLANTIS, the controller is responsible for controlling primitive activities, which are usually reactive sensorimotor processes; the deliberator controls time-consuming computational activities such as planning and world model maintenance; the sequencer coordinates such various activities by initiating and terminating them according to the current goal and situation. Pell et al. [8] proposed a similar architecture for autonomous spacecraft. Such works mainly discuss how to integrate deliberative and reactive activities and deal with the level of planning and executing actions. Since this level corresponds to the decision-theoretic layer in our case, our approach of putting a knowledge-based model selector can also be adopted to these control architectures.

The rest of the paper is organized as follows. Section 2 shows a simple example in which the analysis of the structure of the planning problem is important. Section 3 describes the proposed planning architecture. Section 4 describes the application of the proposed architecture to a mobile robot planning problem in a dynamic environment. Section 5 describes the application to a tactical reasoning in driving used for an intelligent driver assistance system. Section 6 summarizes the paper and discusses future works.

2 Importance of Knowledge of Problem Structure: A Simple Example

This section shows a simple example in which considering the problem structure is important. Figure 2 shows a situation where we are going from city A to city B. We select the route among the three, R_1, R_2, and R_3. The degrees of congestion of R_1 and R_3 are dependent on the current situation, while that of R_2 is constant. We suppose there are two *states*: s_1 is the state that R_1 is more congested than R_3; s_2 is the contrary state. We assume the loss table shown in Table 1, where C_1 and C_2 are losses (it could be the necessary time of travel) imposed by taking a specific combination of the state and the route.

We here suppose that which state actually occurs depends

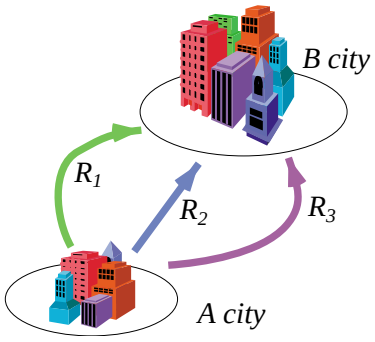


Figure 2. A route selection problem.

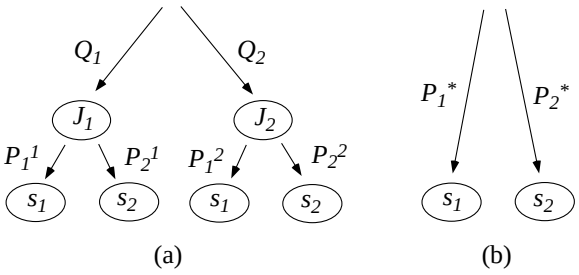


Figure 3. Structure of probabilistic inference.

Table 1. A loss table.

	route to take		
	R_1	R_2	R_3
S_1	C_1	C_2	0
S_2	0	C_2	C_1

on some other factors such as the time of a day. For example, we can consider the case where s_1 is more likely to occur in the morning (we call this situation J_1) and s_2 is in the afternoon (situation J_2). Let P_i^j be the probability that state i occurs in situation J_j and Q_j be the probability of situation J_j (see Figure 3(a)). P_i^j can be regarded as the *model* of situation J_j. If we do not know this hidden structure of the problem (i.e., situation decomposition into J_1 and J_2), we have to use the probabilities of the states directly (see Figure 3(b)), which would probably be estimated from the samples for a whole day. Let P_1^* and P_2^* be such a probability of each state; P_i^* is given by

$$P_i^* = Q_1 P_i^1 + Q_2 P_i^2 \qquad (i = 1, 2)$$

As an example, consider the following parameters: $C_1 = 50$, $C_2 = 20$, $P_1^1 = P_2^2 = 0.8$, $P_2^2 = P_1^2 = 0.2$. Using these values, we summarize in Table 2 the expectation of taking each route in the three cases: (1) the situation is known to be J_1 (i.e., $Q_1 = 1, Q_2 = 0$), (2) the situation is known to be J_2 ($Q_1 = 0, Q_2 = 1$), and (3) the *unknown* situation where J_1

Table 2. Expected loss of taking each route. Underlined is the optimal.

		$E[R_1]$	$E[R_2]$	$E[R_3]$
situation	J_1	40	20	<u>10</u>
	J_2	<u>10</u>	20	40
	unknown	25	<u>20</u>	25

and J_2 are equally likely to occur ($Q_1 = Q_2 = 0.5$). From the table, we can see that if we know the current situation, we can perform planning only for the situation, thereby obtaining a more efficient plan.

If we know the probability of each model, we can select an optimal action, for example, which minimizes the expected loss. Suematsu and Hayashi [16] propose an algorithm to calculate an optimal policy which maximize the expectation of the average reward per step given a set of candidate models and the probabilistic distribution over the set. However such an approach could be computationally expensive, because basically all possibilities (models) have to be examined.

What we would like to stress here are that for efficient (or tractable) model construction and utilization, we should use as much knowledge of the problem structures as possible, and that a planning architecture should be capable of effectively utilizing such knowledge. These are the strong motivation for us to propose the three-level planning architecture.

3 Three-Level Planning Architecture

Based on the above discussion, we propose a three-level control architecture shown in Figure 4. Each level is considered to operate in parallel with the others. The functions of each level is as follows.

Knowledge-based model selector

This level continuously examines the environment and classifies the current situation into one of known categories. Based on the selected category, the corresponding planning model is selected and given to the next level. As a model selector, we can use any classifier; for example, a Bayesian classifier [5] can be used to assess the probability of each category and the reliability of each planning model in the current situation. Concerning the applications presented in this paper, in Section 4, we use a simple frequency-based classifier; in section 5, on the other hand, we use a hand-coded state-transition graph-based classifier.

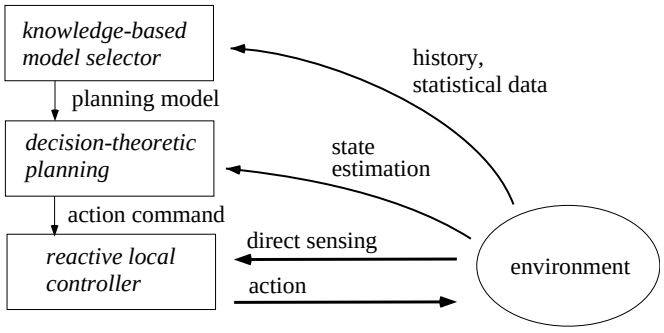


Figure 4. Three-level planning architecture.

Base-level decision-thoretic planner

This level performs planning using the given planning model and the state estimation result to select the best action which minimizes the expected utility, and sends the selected action to the next level. Any decision-theoretic planners can be adopted as long as they response to the dynamics of the environment reasonably quickly. An appropriate example is Real-time Dynamic Programming (RTDP) [1] which on-line generates a decision tree with a limited depth.

Reactive local controller

This level has a set of actions, each of which is realized as a local sensory-action feedback loop. The upper-level decision-theoretic planner selects an action and this level executes it. In addition, this level occasionally handles emergency situations; in that case, this level overrides the upper levels.

4 Example Domain 1: Mobile Robot Planning in Dynamic Environment

This section describes an application of the proposed planning architecture to a mobile robot planning problem in a dynamic environment (see Figure 5).

There is a mobile robot and a moving obstacle. The robot and the obstacle have their own destination and the robot does not know the obstacle's destination. The task of the robot is to reach the destination as early as possible without colliding with the obstacle. The robot has several planning models; only the knowledge of the destination of the obstacle is different in these models. The robot uses one of the models to predict the future movement of the obstacle for decision-theoretic planning.

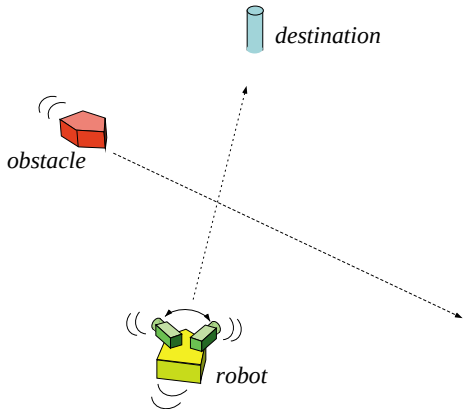


Figure 5. Example problem in dynamic environment.

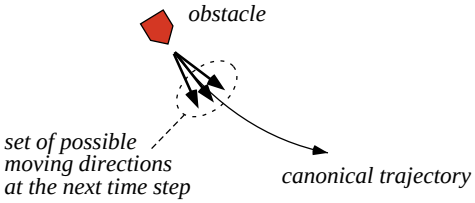


Figure 6. Motion uncertainty model of obstacle.

4.1 Model of Obstacle Motion

Obstacle motion is modeled by its *canonical* trajectory to the destination and motion uncertainty around the trajectory (see Figure 6). We represent the motion uncertainty by a set of possible moving directions at the next time step and the uniform probabilistic distribution over them. We repeatedly apply this uncertainty model to predicting the obstacle position in a near future (a few time steps).

4.2 Decision-Theoretic Robot Motion Selection

Each planning model is composed of the following:

- a state is represented by the position and the velocity of the obstacle and those of the robot.
- an action is the motion (i.e., the moving direction and the speed) of the robot at the next time step.
- a probabilistic distribution is calculated over the possible next position of the obstacle using the motion model (canonical trajectory to a destination).
- a utility function to evaluate the expected time of the robot reaching its destination.

The decision-theoretic planner repeats the cycle of estimating the current state (measuring the position of the obstacle), searching for the best next action, and issuing a command to the controller. The search is performed as follows. First the robot predicts the possible pairs of the position and the velocity of the obstacle and their probabilities at the time a few steps later from the current time by using the motion uncertainty model. For each pair of predicted obstacle position/velocity and robot position, assuming that the obstacle position will diverge around the canonical trajectory (see Figure 7), the robot calculates the time to the destination using our path planner, which generates a minimum-time collision-free path in the time space (see Figure 8). The robot selects the best next action which minimizes the expected time to the destination.

4.3 Selecting Motion Model of Obstacle

The top-level knowledge-based model selector estimates the motion model of the obstacle (i.e., planning model) from the history of obstacle motion. At present, we have tested the following two types of model selectors.

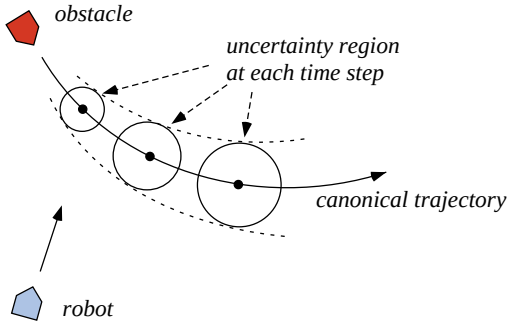


Figure 7. Uncertainty evolution model of obstacle motion.

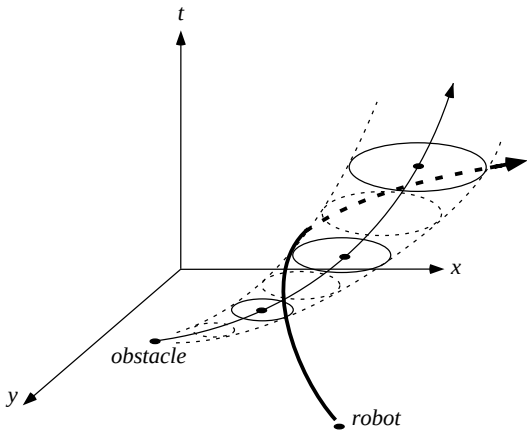


Figure 8. Path planning in time-space.

4.3.1 Static Model Selector

This is a very simple frequency-based selector which does not consider much about the dynamics of the environment. Let M_i be models and P_i be their probabilities. Each probability is calculated from the relative frequency in the latest N_h trials.[1] Let n_i be the frequency of the ith model in the trials. The probability P_i is given by

$$P_i = \frac{n_i}{N_h}.$$

Let i^* be the index of the most probable model. If P_{i^*} is above a predetermined threshold, the i^*th model is selected; otherwise, the model selector considers that the obstacle does not have no canonical trajectory and considers all possible moving directions of the obstacle derived from all possible candidates for the canonical trajectory. These two cases are analogous to the case in Section 2 where the situation is known to be J_1 or J_2 and the case where the situation is unknown, respectively.

[1] We assume that, after each trial, the robot can uniquely determine the motion model of the obstacle during the trial.

4.3.2 *Dynamic Model Selector*

This is also a simple frequency-based selector, but it investigates the underlyning dynamics of the environment (i.e., moving obstacle). That is, the selector considers the change of the moving obstacle's destination as a Markov process and estimates the transition matrix M of the Markov process from the history of obstacle motion. The element $M_{i,j}$ of the matrix indicates the probability that the obstacle that took the ith model at the latest trial takes the jth model at the next trial, and is estimated by:

$$M_{i,j} = \frac{n_{i,j}}{N_i}, \tag{1}$$

where $n_{i,j}$ is the frequency of the jth model taken just after the ith model and N_i is the frequency of the ith model.

4.4 Simulation Results

Figure 9 shows the problem setting for simulation. The robot moves upward and the obstacle moves downward. There are three motion models of the obstacle, which are referred to as LW (leftward, from the robot's point of view), ST (straight), and RW (rightward). A canonical trajectory of the obstacle is calculated for each pair of the current and the goal position. The robot classifies the situation into one of these three models.

In the simulation, we can consider two kinds of motion models of the obstacle: one is the model that the robot expects for the obstacle (*expected model*); the other is the model that the obstacle actually takes (*actual model*). If these two models coincide, the robot motion is expected to be efficient; otherwise, the robot is likely to exhibit an inefficient behavior. Figure 10 shows two exmaples of trials. In Figure 10(a), the robot thought the obstacle would move leftward (LW) and the obstacle actually moved as expected (LW); in the Figure 10(b), on the other hand, although the robot thought the obstacle would move rightward (RW), the obstacle actually moved leftward (LW). The robot motion is much more efficient in the first case than in the second.

Table 3 shows the result of simulation trials to see how the relationship between the expected and the actual models affects to the performance of the robot. In the table, each

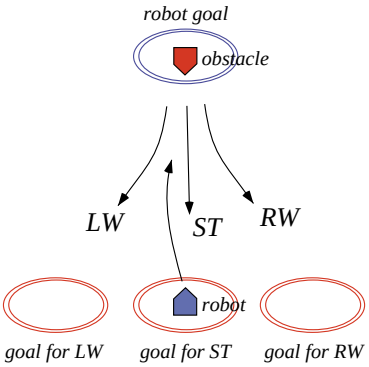
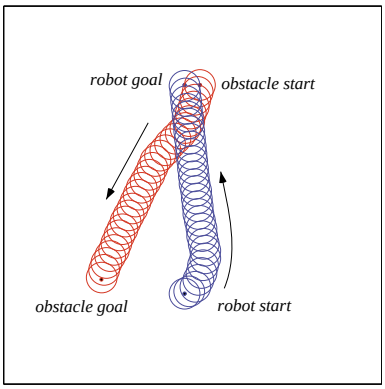
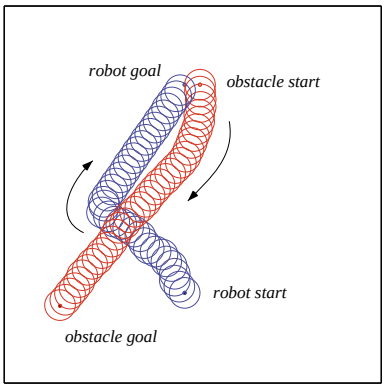


Figure 9. Problem setting.



(a) expected=LW, actual=LW.



(a) expected=LW, actual=RW.

Figure 10. Results of two trials.

Table 3. Simulation results.

		ACTUAL			
		NO	*ST*	*LW*	*RW*
E	*NO*	32.55	34.59	33.18	33.31
X		(6.57)	(2.58)	(4.39)	(5.05)
P	*ST*	38.21	29.63	31.26	35.34
E		(12.14)	(1.02)	(3.49)	(6.82)
C	*LW*	37.01	31.35	29.72	34.39
T		(8.49)	(1.81)	(2.34)	(5.71)
E	*RW*	32.97	31.41	32.89	29.71
D		(6.16)	(1.82)	(2.73)	(2.62)

row indicates the expected model and each column indicates the actual model. The NO-row indicates that the robot has no models of obstacle motion; the NO-column indicates that the obstacle randomly selects one of the three models (LW, ST, RW). For each combination of the two models, we ran 150 trials. In each box, the upper number is the mean of the time steps the robot took to reach the destination. The lower number in parentheses is the standard deviation of the time steps. We can see from the table that the robot motion (and equivalently the planning result) is efficient if an appropriate planning model is selected, and it is not otherwise.

Next we tested the three-level planning architecture through a large number of concecutive trials. The knowledge-based

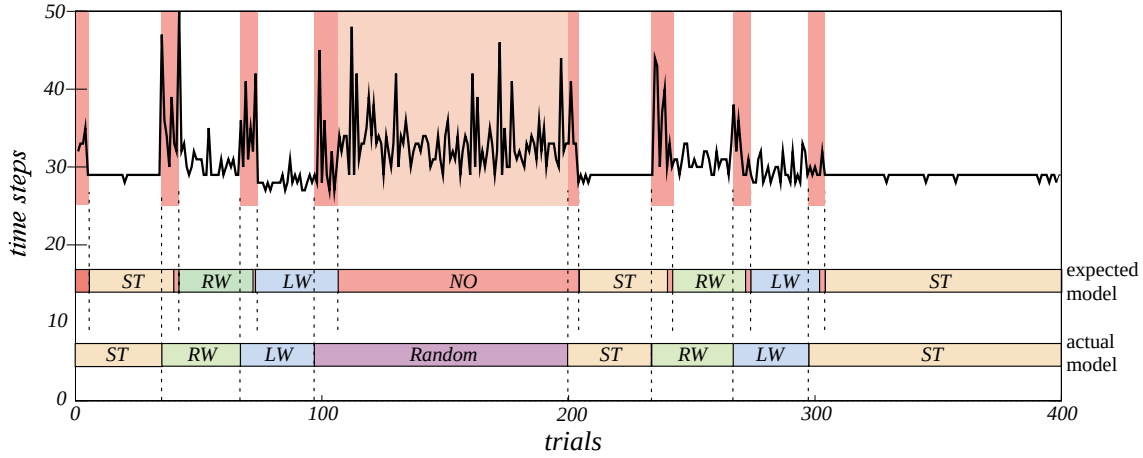


Figure 11. obstacle's actual behavior, expected behavior, and elapsed time steps.

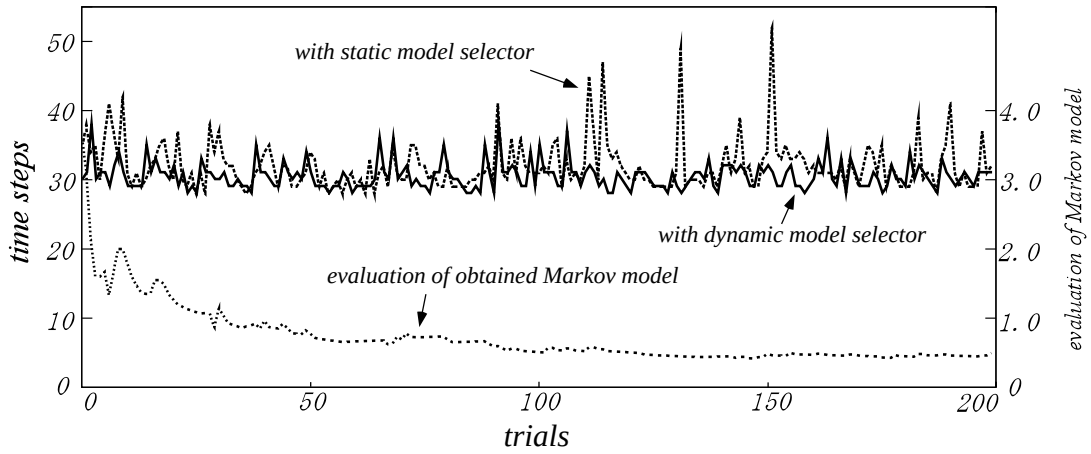


Figure 13. Comparison of planners with dynamic and static model selector.

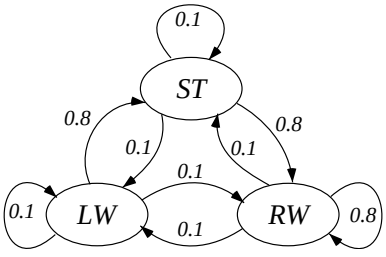


Figure 12. Markov process used.

model selector collects the history of the obstacle movement and determines the current model based on the probability estimation method mentioned above.

The first test was done with the static model selector (see Sec. 4.3.1). Figure 11 shows the result. The figure shows the time elapsed by the robot to reach the destination (upper part), the expected model of the obstacle (middle part), and the actual model of the obstacle (lower part). In the upper part, the ranges of trials where there was a discrepancy between the expected and the actual model are shown as shaded regions. For the NO actual model, the NO is the best expected model, but in this case, we can say that there is always a discrepancy; so the range corresponding to such a case is also lightly shaded. In such shaded regions, the robot motion is not efficient due to generating a plan based on an inappropriate planning model.

The second test was done with the dynamic model selector (see Sec. 4.3.2). In this case, we used the Markov process shown in Figure 12 as the underlyning dynamics to generate the problem sequence. In addition, the *expected* motion model

is selected so that the predicted expectation of the cost is minimized using the predicted probability of each model and also using Table 3 as the expected cost for each combination of the actual and the predicted model. Figure 13 shows the result. In the figure, the performance of the planning system with the static model selector and that with the dynamic model selector as well as the change of the evaluation of the estimated Markov model[2]in the dynamic model selector are shown. The result shows the dynamic model selector outperforms the static one because the dynamic model selector utilizes the knowledge about the problem structure that the obstacle changes the motion model according to some Markov process.

5 Example Domain 2: Intelligent Driver Assistance

This section briefly describes an application of the three-level planning architecture to the *intelligent navigator* that can give the driver timely advice on safe and efficient driving. For the details of the intelligent navigator, refer to [12].

Usually tasks in driving can be divided into two levels [17]: the *tactical level* determines maneuvers such as lane changing and overtaking to meet the objective of driving (e.g., a target arrival time) under the constraints imposed by the actual traffic condition; the *operational level* translates the selected maneuver into actual operations of steering, accelerating, adn braking.

In real traffic, sensory data based on which the intelligent navigator generates advice is *uncertain* (e.g., measurement error or occlusion). In addition, the situation is *dynamic*, i.e., the situation evolves as time elapses. Therefore, the tactical level advice generation should be based on the prediction of the future traffic condition with consideration of uncertainty. We adopt a decision-theoretic planning for the tactical level.

Then, in order to adaptively select a planning model for the tactical level and to activate the tactical level only when it is necessary, we introduce a meta-level planning (called the *meta-tactical level*). The resultant control architecture is composed of three levels: *meta-tactical level*, *tactical level*, and *operational level*; they exactly correspond to the three levels shown in Figure 4.

5.1 Tactical Level Decision-Thoretic Planning

Figure 14 shows a scenario where a decision-theoretic planning is employed to determine a maneuver. Our vehicle (painted rectangle) is on the left lane [3], and is approaching the exit to take. Since the speed in the current lane is becoming a little bit slow, the driver starts thinking of overtaking vehicles ahead. The overtaking maneuver is generally faster, but there may be risks of lane changing itself and of missing the exit due to occluded vehicles ahead. Such a trade-off is formalized in a *planning model* corrsponding to each situation.

In this application, each planning model is composed of the following:

[2] The model is evaluated by the sum of absolute differences of the corresponding elements of the obtained transition matrix and the true matrix.

[3] Note that the slower lane is the left one in Japan.

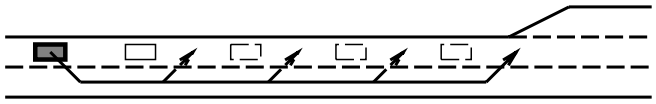


Figure 14. An overtaking scenario.

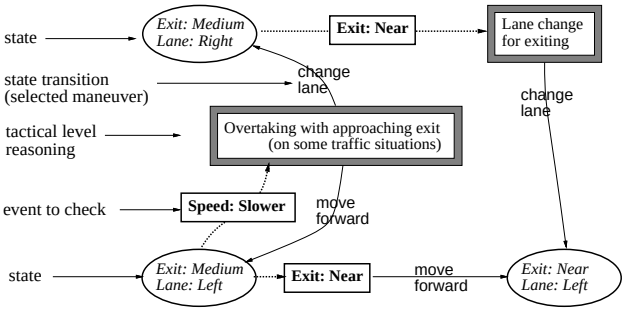


Figure 15. A part of state-transition graph for the meta-tactical level.

- a state is represented by the position and the velocity of our vehicle and those of other vehicles.
- an action is a maneuver. In the above scenario, there are two actions: *change lane for overtaking* and *go straight*.
- a probability distribution is calculated over the possible future placements of vehicles including occluded ones.
- a utility function which is a function of the expected time of our vehicle reaching the exit and the loss of each maneuver.

We also have constructed planning models for other traffic situations such as: overtaking near the target exit without congestion; congestion around an entrance or an exit which our vehicle does not take. All such models are manually constrcuted.

5.2 Meta-Tactical Level as Knowledge-Based Model Selector

This level continusouly watches important events on traffic. Examples of possible events are: the average speed of the current lane slows down; the exit is approaching. It also periodically updates the estimation of the arrival time. Since it is inefficient to always check all events, only selected events are monitored which are considered to be important in the current state. To realize such an adaptive focus of attention, we construct a state transition graph. Figure 15 shows a part of the transition graph. For example, at state `[Exit: Medium, Lane: Left]` (which means that the distance to the exit is medium and the vehicle is on the left lane), possible events are: (1) the speed becomes slower (`Speed: Slower`); (2) the exit becomes near (`Exit: Near`). For each event, the corresponding planning model for the tactical level is assigned. This strcuture enables the meta-tactical level to adaptively select appropriate planning models.

5.3 Implementation and Experiments

The advice generation subsystem with the three-level architecture is connected to a road scene visual recognition subsystem, which detects and localizes lanes and other vehicles, to constitute the intelligent navigator. We constructed a prototype system and conducted experiments on an actual highway. In one case, for example, our vehicle with the intelligent navigator traveled about 12km, and during the travel, the intelligent navigator generated advice 5 times, all on appropriate timings.

6 Conclusions and Discussion

This paper has discussed the importance of analyzing the problem structure and of the model selection mechanism for an agent making a plan in uncertain and dynamic environments. A three-level planning architecture has been proposed which has a model selection layer on top of an decision-theoretic middle layer. We applied the architecture to two planning problems. In a mobile robot planning in a dynamic environment, the model selector is implemented as a frequency-based model estimator, which can select an appropriate planning model adaptively. Model selection by the dynamic model selector is done based on the expected loss due to model discrepancy. In the intelligent navigator example, the model selector is implemented as a state-transition graph, which selects planning models according to both the history of maneuvers and the current traffic condition. These two application examples show the generality and the usefulness of the proposed architecture.

This paper has focused only on the use of the knowledge-based meta-level control for planning model selection. Another important role of the meta-level control is to limit the search space adaptively according to the time pressure. The following two approaches are possible: limiting the set of action candidates and limiting the length of lookahead. These kinds of meta-level control could be realized by a method which explicitly considers the tradeoff between the amount of search and the plan quality. However, formulating such a tradeoff could sometimes be difficult in complex planning problems in the real world. Therefore, our approach of employing knowledge-based control seems to have an advantage in a practical use.

In this paper, we have manually constructed the meta-level model selector by examining the problem structure. A future work is to automate the construction of the model selector to some extent, by using various machine learning techniques.

REFERENCES

[1] A.G. Barto, S.J. Bradtke, and S.P. Singh, 'Learning to act using real-time dynamic programming', *Artificial Intelligence*, **72**(1-2), 81–138, (1995).

[2] J. Blythe, 'Decision-theoretic planning', *AI Magazine*, **20**(2), 37–54, (1999).

[3] M. Boddy and T. Dean, 'Solving time-dependent planning problems', in *Proceedings of IJCAI-89*, pp. 979–984, (1989).

[4] C. Boutilier, R. Dearden, and M. Goldszmidt, 'Exploiting structure in policy construction', in *Proceedings of the Fourteenth Int. Joint Conf. on Artificial Intelligence*, pp. 1104–1111, (1995).

[5] W.L. Buntine, 'Operations for learning with graphical models', *J. of Artificial Intelligence Research*, **2**, 159–225, (1994).

[6] A. Cameron and H. Durrant-Whyte, 'A bayesian approach to optimal sensor placement', *Int. J. of Robotics Res.*, **9**, 70–88, (1990).

[7] T. Dean, L.P. Kaelbling, J. Kirman, and A. Nicholson, 'Planning with deadlines in stochastic domain', in *Proceedings of AAAI-93*, pp. 574–579, (1993).

[8] B. Pell et al., 'An autonomous spacecraft agent prototype', in *Proceedings of Agents-97*, (1997).

[9] E. Gat, 'Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots', in *Proceedings of AAAI-92*, pp. 809–815, (1992).

[10] E.J. Horvitz, *Computation and Action Under Bounded Resources*, Ph.D. dissertation, Stanford University, 1990.

[11] S.A. Hutchinson and A.C. Kak, 'Planning sensing strategies in a robot work cell with multi-sensor capabilities', *IEEE Trans. on Robotics and Automat.*, **5**(6), 765–783, (1989).

[12] M. Itoh, J. Miura, and Y. Shirai, 'Towards intelligent navigator that can provide timely advice on safe and efficient driving', in *Proceedings of the 1999 IEEE/IEEJ/JSAI Int. Conf. on Intelligent Transportation Systems*, pp. 981–986, Tokyo, Japan, (October 1999).

[13] J. Miura and Y. Shirai, 'Vision and motion planning for a mobile robot under uncertainty', *Int. J. of Robotics Research*, **16**(6), 806–825, (1997).

[14] J. Miura and Y. Shirai, 'Vision-motion planning for a mobile robot considering vision uncertainty and planning cost', in *Proceedings of the 15th Int. Joint Conf. on Artificial Intelligence*, pp. 1194–1200, Nagoya, Japan, (August 1997).

[15] S. Russell and E. Wefald, *Do The Right Thing*, The MIT Press, 1991.

[16] N. Suematsu and A. Hayashi, 'Consideration of model uncertainty in decision theoretic planning', *Trans. Information Processing Soc. of Japan*, **40**(7), (1999). (in Japanese).

[17] R. Sukthankar, S. Baluja, J. Hancock, D. Pomerleau, and C. Thorpe, 'Adaptive intelligent vehicle modules for tactical driving', in *1996 AAAI Workshop on Intelligent Adaptive Agents*, pp. 13–22, (1996).