Generating Optimal Policies for High-level Plans with Conditional Branches and Loops

Shieu-Hong Lin Thomas Dean[1]
Brown University
Department of Computer Science
115 Waterman Street
Providence, RI 02906, USA
Phone: (401) 863-7600
Fax: (401) 863-7657
Email: {shl,tld}@cs.brown.edu

Abstract

We are concerned with generating optimal policies for Markov decision processes that are represented as high-level plans with conditional branches and loops. Often complex planning processes can be broken down into elementary plan steps with associated restricted sets of actions. These plan steps can be combined to form high-level plans using a simple programming language specifying conditionals, loops, and sequences involving the plan steps as primitive statements. It is infeasible to directly generate and solve the underlying Markov decision process, since the size of the state space is exponential in the size of a high-level plan. We address the problem of efficiently computing an optimal policy by taking advantage of locality structure in the high-level plan. The main result is the specification and analysis of an algorithm that takes as input a high-level plan and provides as output an optimal policy for the underlying Markov decision process.

Keywords: planning, action representation, uncertainty, stochastic domains, decision-theoretic planning, Markov decision processes

1. Introduction

Reachability in state space greatly affects the computational cost of planning and decision making. By disabling some actions and enabling others in different steps of the decision making process, reachability in the search space can be considerably localized. The work on supervisory controllers in the study of discrete event systems is predicated on this basic idea [9]. For example, if I tell you to travel to Chicago from Providence, the search space for constructing a travel plan is quite large. If in addition I tell you to accomplish this task by first traveling to Logan Airport in Boston using one of two airport shuttle services and then flying to O'Hare Airport in Chicago using either United or Delta Airlines, then the search space is considerably smaller.

[1]This work was supported in part by a National Science Foundation Presidential Young Investigator Award IRI-8957601, by the Air Force and the Advanced Research Projects Agency of the Department of Defense under Contract No. F30602-91-C-0041, and by the National Science foundation in conjunction with the Advanced Research Projects Agency of the Department of Defense under Contract No. IRI-8905436.

Often complex decision processes can be broken down into such modular plan steps (*e.g.,* travel to Logan Airport) associated with a restricted set of actions (*e.g.,* use shuttle service A or B). These plan steps can be combined into high-level plans (*e.g.,* travel to Logan, fly to O'Hare) using simple programming languages that allow the specification of conditionals, loops, and sequences involving the plan steps as primitive statements. By adding conditionals and loops, the decision process mentioned above can be further complicated. For example, on good-weather days we travel to San Francisco through Chicago, but on bad-weather days we head for Washington, D.C., and then try repeatedly to get a direct flight to either San Francisco or nearby San Jose. Appropriate languages for specifying loops and conditional include Petri nets and more restricted task network formalisms such as the one proposed in [10].

In this paper, we assume that for each initial state and action possible in that state we have a probability distribution governing the next state, and a cost function indicating the cost incurred for each possible next state. These probability distributions and cost functions have the Markov property that they are independent of the history prior to the initial state. For example, in snowy weather, shuttle service A may fail to get you to the airport to catch your flight with probability 0.05, while the probability of failure is only 0.02 if there is no snow; however, shuttle service A charges \$20 only if you arrive in time for your flight. On the other hand, shuttle service B may successfully convey you to the airport with probability 0.99 no matter what the weather is, but shuttle service B charges \$30 whether or not you arrive in time for your flight. A natural question to ask is under what circumstances should you use shuttle service B instead of A. More generally, given a high-level plan, we would like to determine an optimal policy such that we can (i) map observed states in each individual plan step to actions enabled in the plan step, and (ii) minimizes the expected cumulative cost.

A high-level plan as described above represents a Markov decision process [10]. There exist standard techniques that can determine an optimal policy for a Markov decision process in time polynomial in the size of the given state space [7] [8]. However, the size of the state space of the underlying Markov decision process is exponential in the number of state variables appearing in the high-level plan. It is infeasible to directly apply the standard techniques to a Markov decision problem represented by the kind of high-level plans considered in this paper.

Our goal is to expedite the computation of an optimal policy for the underlying Markov decision process by taking advantage of locality structure embedded in the high-level plan. First of all, it is likely that not all state variables are involved in all portions of the high-level plans. For different planning steps, only restricted sets of actions are considered. By disabling some actions and enabling others, reachability in the state space is considerably limited. In addition, the effects of an action tend to depend on the values of a small number of state variables prior to taking the action, and only the values of a small number of variables are affected after the action. Furthermore, it is rare that the plan steps are executed with uniform frequencies as time evolves. Instead, we tend to loop in a fragment of the plan for a period of time, and then move on to anther fragment. By exploiting the restricted reachability in time and state space described above, we are able to consider state subspaces of reduced dimensionality for these fragments separately, instead of the entire original state space.

The main result in this paper is the specification and analysis of an algorithm that takes as input a high-level plan and provides as output an optimal policy for the underlying Markov decision process. The algorithm is loosely based on the work of Lin and Dean [6] for answering queries in temporal reasoning problems. Assuming a reasonable amount of locality embedded in a high-level plan, our analysis shows that this algorithm can compute an optimal policy in time polynomial in the size of the high-level plan. The remainder of this paper is organized as follows. In Section 2, we formally describe the kind of high-level plans considered in this paper and the underlying Markov decision processes. Section 3 characterizes a particular family of plan fragments. Section 4 describes the dependencies among plan fragments and the state variables, and the abstracted state subspaces of an individual plan fragment. Section 5 depicts the construction of local Markov decision processes for the family of plan fragments described in Section 3. Section 6 introduces a dynamic-programming algorithm that derives an optimal policy for the high-level plan by systematically solving the local Markov decision processes described in Section 5. Section 7 contains the analysis of the algorithm and Section 8 briefly mentions some related work.

2. High-level Plans: Representation and Optimization

2.1. Domain Dynamics and Locality

We assume that the dynamics for the domain is represented by $\langle V, \Omega_A, \text{Space}(V), \Pr, \text{Cost}\rangle$ as follows:

- $V = \{X_1, \ldots, X_n\}$ is a set of discrete state variables. Ω_{X_i} is the set of possible values for X_i. $\text{Space}(V) = \prod_{i=1}^{n} \Omega_{X_i}$ is the state space determined by V.
- Ω_A is a set of actions. For the actions in Ω_A, Pr and Cost specify the conditional probabilities and the costs associated with actions and action outcomes.

Let $X_{i,t}$ denote a random variable representing the state variable X_i at time t. In the most general form, the conditional probability associated with an action k in Ω_A,

$$\Pr(X_{1,t}, \ldots, X_{n,t} | k, X_{1,t-1}, \ldots, X_{n,t-1}),$$

specifies for each given state at time $t-1$, the probability distribution over the possible states at time t if action k is carried out at time $t-1$. Similarly the cost,

$$\text{Cost}(X_{1,t}, \ldots, X_{n,t} | k, X_{1,t-1}, \ldots, X_{n,t-1}),$$

indicates for each given state in time $t-1$, the cost of ending up in a specific state at time t if action k is carried out at time $t-1$.

Locality in the cause-and-effect relationship can enable us to represent the costs and conditional probabilities associated with actions and action outcomes more compactly as probabilistic state-space operators as in [4] or as temporal probabilistic networks as in [2]. Typically, (i) an action k in Ω_A only affects the values of the variables in a small subset $Y_k = \{Y_1^k, \ldots, Y_b^k\}$ of V, according to the values of a small subset

$Z_k = \{Z_1^k, \ldots, Z_c^k\}$ of V prior to taking action k, and (ii) the cost of taking action k only depends on the values of a small subset $T_k = \{T_1^k, \ldots, T_d^k\}$ of V immediately prior to action k, and a small subset $U_k = \{U_1^k, \ldots, U_e^k\}$ of V immediately following action k We define $V_k = Y_k \cup Z_k \cup T_k \cup U_k$ as the set of *relevant* state variables for action k. By ignoring the irrelevant state variables, we can more compactly represent the conditional probability and the cost function associated with action k as

$$\Pr(Y_{1,t}^k, \ldots, Y_{b,t}^k | k, Z_{1,t-1}^k, \ldots, Z_{c,t-1}^k),$$

$$\text{Cost}(U_{1,t}^k, \ldots, U_{e,t}^k | k, T_{1,t-1}^k, \ldots, T_{d,t-1}^k).$$

In this paper, we are concerned with exploiting locality in action dynamics given that only small subsets of the variables in V, instead of the whole set of variables in V, are relevant to individual actions.

2.2. High-level Plans: Representation and Optimization

A high-level plan is composed of a set of interconnected *plan steps* and *control nodes*. In the following, we formally describe the representation, the execution, and the optimization of high-level plans.

Plan steps: A plan step $R = (G_R, K_R, V_R)$ is composed of (i) a guard expression G_R, (ii) a set of actions K_R, $K_R \subset \Omega_A$, and (iii) a set of the state variables V_R, $V_R \subset V$. G_R is a boolean expression concerning the values of a subset of the variables in V, which must be true immediately prior to entering plan step R. K_R is the set of actions enabled in plan step R. All activity within plan step R is restricted to the actions in K_R. V_R is the set of state variables *appearing* in plan step R, which is the union of the state variables appearing in G_R and the state variables directly relevant to the actions in K_R.

Control nodes: A control node $C = (G_C)$ is composed of a guard expression G_C, which is a boolean expression concerning the values of some variables in V. A control node serves as a valve to control the execution of a plan. The guard expression G_C must be true before we can bypass a control node C.

High-level plans: A high-level plan H is represented as a directed graph $H = (N_H, E_H)$ where (i) each node R in N_H is either a plan step or a control node and (ii) a directed edge (R, Q) in graph E_H indicates that after taking an action enabled in a plan step R (or after bypassing a control node R), we can enter a plan step Q (or bypass a control node Q) if and only if the guard expression G_Q is true.

Relationship among plan steps: A plan step Q is a *child* of a plan step R (or R is a *parent* of Q) if and only if R can reach Q by passing zero or more control nodes in E_H We then naturally extend this definition to define the the *ancestors* and *descendants* of plan steps in an obvious way.

The starting node and the ending node of a plan: A starting (ending) node is a control node that is the unique source (sink) in $H = (N_H, E_H)$. We assume that there

is a unique source (sink) since we can always construct one by adding a dummy control node that leads to (follows) all the current sources (sinks).

Plan execution and uncertainty: A plan execution starts at the starting node, and then branches and loops around individual plan steps until we enter the ending node [2]. Different executions of the same plan can follow different paths of plan steps due to (i) the various choices of actions in individual plan steps and (ii) the uncertainty in the action outcomes.

Loops and conditional branching in plan execution: The guard expressions of plan steps and control nodes direct the execution of a plan. We can enter a plan step R (or bypass a control node R) if and only if its guard expression G_R is true immediately prior to entering R (bypassing R). Each plan step R may have several parents and children. We repeatedly take actions in a plan step R until we can successfully bypass zero or more control nodes and enter a child plan step Q of R [3]. We then branch into the execution of Q.

Action costs and final reward: Each time we take an action during a plan execution, the action involves a cost as described in Section 2.1. At the end of a plan execution, we get a final reward determined by a linear function $\sum_{1 \leq i \leq n} r_i X_i$ regarding the values of the state variables at the end.

Optimizing a high-level plan: A policy is a mapping from states and plan steps to actions. Given the current state and the current plan step in a plan execution, a policy determines which action to take. Our task is to find an optimal policy that minimizes the expected cumulative action cost minus the expected final reward in executing a plan.

2.3. An Example: a Quality Improvement Plan

Figure 1 shows a high-level plan to improve the overall quality of a product with ten components $X_1, \ldots, X_{10}$. The quality of each X_i is quantified as a value that is either zero or one. Initially the values of all of the components are rated as zero. At the end of the improvement, we receive a reward proportional to the number of the components X_i's whose values are rated as one. In addition, it is also required that the values of at least (i) one of $\{X_1, X_2\}$, (ii) X_4 and two of $\{X_3, X_5, X_6\}$, and (iii) three of $\{X_7, X_8, X_9, X_{10}\}$ must be rated as one at the end of the improvement.

Figure 1 displays thirteen plan steps and control nodes connected as a directed graph, where (i) C_{start} and C_{end} are the starting node and the ending node respectively, (ii) C_{start}, C_{end}, C_1, C_2, and C_3 are all control nodes to direct the execution of the plan, and (iii) $S_1, \ldots, S_8$ are the plan steps that allow us to change the quality of the product by taking actions. Each of the plan steps in $\{S_1, \ldots, S_8\}$ is associated with two enabled actions. Taking an action enabled in a plan step incurs a cost and changes the values of

[2] We focus on high-level plans that with probability 1 end in the ending node in the long run.

[3] We assume that no more than one of the child plan step can be entered at a time.

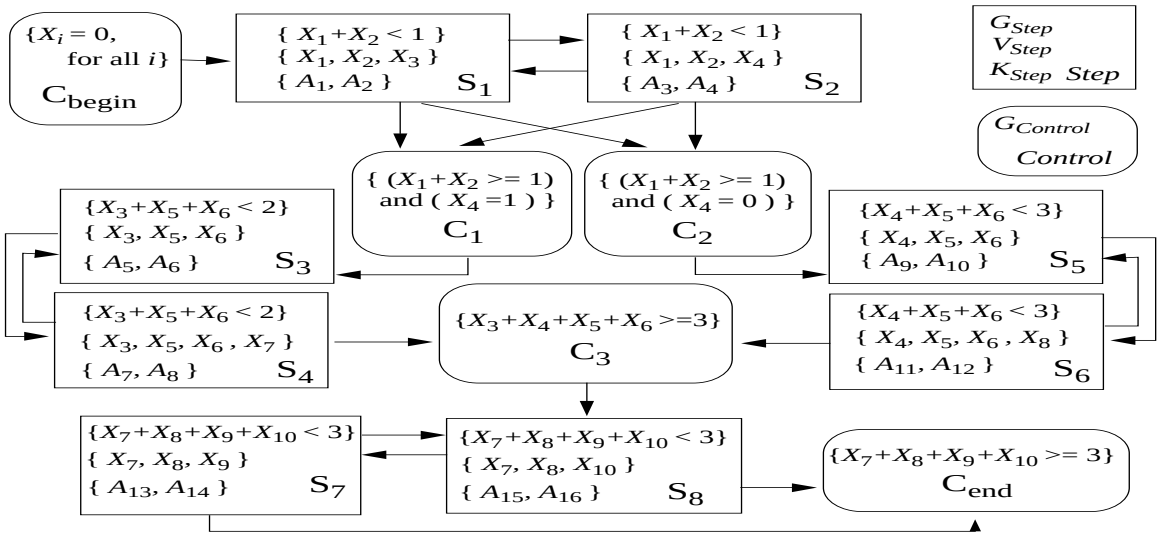

Figure 1: A quality improvement plan

Figure 1: A high-level plan to improve product quality

the components X_i's in a nondeterministic fashion as described in Section 2.1. Rather than providing the details of the conditional probabilities and the costs associated with actions and action outcomes, we simply depict the state variables appearing in the individual plan steps in Figure 1.

The expected cumulative cost and the expected final reward of in executing a plan are greatly affected by how we take actions in individual plan steps. For example, in step S_1 (i) action A_1 may improve the values of at least one of X_1 and X_2 to one with cost \$15 and probability 0.75 all the time, while (ii) action A_2 may improve the values of both X_1 and X_2 to one with cost \$20 and probability 0.99 when the value of X_4 is one immediately prior to taking action A_2. Instead of taking action A_1 in step S_1 all the time, it may be worthwhile to take action A_2 if the value X_4 is one. A policy of such a plan is a mapping that for each plan step R in $\{S_1, \ldots, S_8\}$ determines the action to take according to the values of the components immediately prior to taking an action enabled in R. Our task is to derive an optimal policy, which minimizes (i) the expected cumulative cost of actions during plan execution minus (ii) the expected reward at the end of plan execution.

3. Locality in Plan Execution

Given a high-level plan H, the size of the state space Space(V) is exponential in the number of the state variables in V. It is infeasible to explore such a state space to determine an optimal policy for the high-level plan H. In this section, we investigate locality embedded in high-level plans. This enables us to derive an optimal policy by considering separate state subspaces of reduced dimensionality for different portions of the plan.

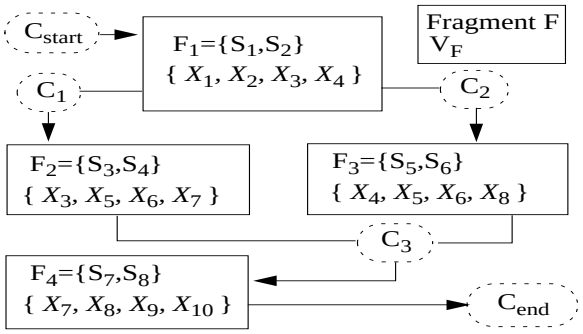

Figure 2: The fragment graph of the quality improvement plan

3.1. Coherent Fragments and Locality

In the following, we formally define the coherent fragments of a high-level plan and indicate the underlying locality property.

Fragments and coherent fragments: Given a high-level plan represented as a graph $H = (N_H, E_H)$, a *fragment* F of the plan is a subset of the plan steps in N_H. A fragment F of H is a *coherent* fragment [4] if and only if F is a maximal fragment in H where all the plan steps in F are directly or indirectly connected to one another. *Remark:* In other words, a coherent fragment is composed of the plan steps in a strongly-connected component of the graph $H = (N_H, E_H)$. Therefore coherent fragments are always disjoint.

The set of variables appearing in a fragment F, V_F: A variable X_i is in V_F if and only if X_i appears in at least one of the plan steps in F.

Relationship between coherent fragments: Given two coherent fragments F and G, we say that G is a child of F if and only if at least one of the plan steps in F has a child plan step in G. Naturally, we can also define descendant fragments, parent fragments, and ancestor fragments in a similar way. For a coherent fragment F, Children(F), Descendants(F), Parents(F), and Ancestors(F) denote the sets of the coherent fragments that are the children, the descendants, the parents, and the ancestors of F respectively.

The fragment graph of a high-level plan: The fragment graph $(\mathcal{F}, E_{\mathcal{F}})$ of a high-level plan H is a directed graph where (i) $\mathcal{F}$ is composed of all of the coherent fragments in H, and (ii) (F, F') is a directed edge in $E_{\mathcal{F}}$ if and only if F' is a child coherent fragment of F. A fragment graph indicates the relationship between coherent fragments.

Figure 2 depicts the coherent fragments, the variables appearing in coherent fragments, and the fragment graph of the quality improvement plan in Figure 2. For clarity, on the directed edges of the fragment graph we also label the control nodes that join adjacent fragments. We are interested in exploiting the following locality property of coherent fragments during plan execution:

Property 1 *The fragment graph of a high-level plan is a directed acyclic graph. As soon as we enter a coherent fragment F for the first time, we branch and loop in the plan steps in fragment F before we enter another coherent fragment. After we leave*

[4] In the remainder of this paper, we focus on coherent fragments.

fragment F, we never visit F again.

Example: Consider the quality improvement plan in Figure 1. There are three stages in a plan execution with respect to the three requirements on the final values of the components.

First, we (i) start a plan execution in control node C_{start}, (ii) enter coherent fragment F_1 through C_{start}, and (iii) alternate between steps S_1 and S_2 to improve the values of at least one of X_1 and X_2 to one. As soon as this requirement is satisfied, we transfer to the second stage. (Note that the values of X_1 and X_2 are not affected after this stage.)

At the second stage, if the value of X_4 has been improved to one at the end of stage one, we (i) enter coherent fragment F_2 through control node C_1 and (ii) alternate between steps S_3 and S_4 to improve the values of at least two of X_3, X_5 and X_6 to one. Otherwise, we (i) enter coherent fragment F_3 through control node C_2 and (ii) alternate between steps S_5 and S_6 to improve the values of all of X_4, X_5 and X_6 to one. In either case, we transfer to the third stage as soon as the second requirement is satisfied. (Note that the value of X_4, X_4, X_5 and X_6 are not affected after this stage.)

At the third stage, we (i) enter coherent fragment F_4 through control node C_3, (ii) alternate between steps S_7 and S_8 in to improve the values of at least three of X_7, X_8, X_9 and X_{10} to one, and (iii) then finish the entire plan execution in control node C_{end}.

3.2. Dependency and Locality

In many situations, not all variables are directly relevant throughout an entire plan execution. Instead, variables only directly participate in a few coherent fragments, and then propagate its effects through other variables to the rest of the plan. Consider the quality improvement plan in Figure 1. Variables X_1 and X_2 only appear in fragment F_1. At the end of the execution of F_1, the effects of X_1 and X_2 in F_1 are then propagated to fragments F_2 and F_3 through the variables X_3 and X_4 respectively. Similar situations occur in the other coherent fragments too.

In these situations, coherent fragments are loosely coupled: (i) only a few state variables in V appear in a coherent fragment, and (ii) a coherent fragment interacts with its child coherent fragments only through a few state variables. In the following, we identify for each coherent fragment F [5] (i) the set of state variables actively involved in F, and (ii) given a child fragment G of F, the sets of the state variables that characterize the interactions between F and G.

Figure 3 displays these sets of variables that characterize the causal dependencies among the coherent fragments in the quality improvement plan.

The set of active variables in F, A_F: A variable X_i is in A_F if and only if (i) X_i appears in at least one of F or F's ancestor fragments, and (ii) X_i appears in at least one of F's or F's descendant fragments. The active variables in F are the direct media that (i) propagate the effects of the plan execution in F's ancestor fragments to F or (ii) propagate the effects of the plan execution in F to the descendant fragments of F. Contrastingly, a variable X_j is a *passive variable* in F if and only if either (i) X_j does not appear in any of F or F's ancestor fragments, *or* (ii) X_j does not appear in any of

[5] We mean a coherent fragment whenever we use the term "fragment" in the following discussion.

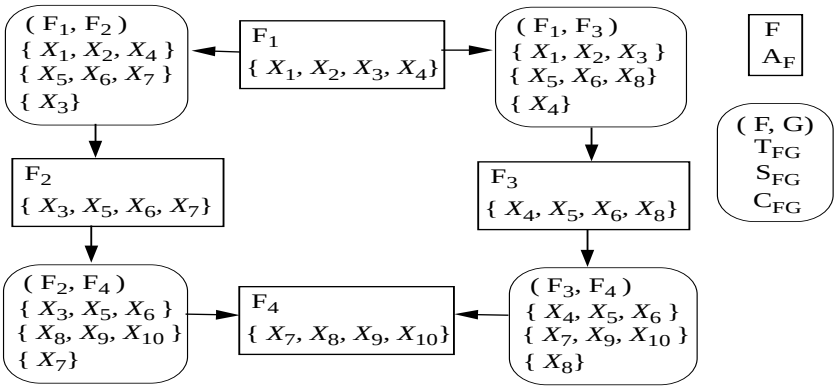

Figure 3: The causal dependencies among state variables and coherent fragments

Figure 3: Causal dependencies among coherent fragments

F or F's descendant fragments. In both cases, the information about X_j is redundant during the plan execution in F.

Property 2 *Only the active variables in F are directly relevant to the plan execution in F. A passive variable in a coherent fragment F can only indirectly affect or be affected by the plan execution in F through the set of active variables in F.*

The set of terminal variables when leaving F to enter G, T_{FG}: A variable X_i is in T_{FG} if and only if (i) X_i appears in F or one of F's descendant fragments, and (ii) X_i does not appear in any of G or G's descendant fragments.
Remark: A terminal variable X_i in T_{FG} is an active variable in F but is no longer an active variable after entering G. X_i can be abstracted away and receive a reward $r_i X_i$ in advance when we leave F to enter G, since the value of X_i is never changed after that point.

The set of starting variables when entering G from F, S_{FG}: A variable X_i is in S_{FG} if and only if (i) X_i does not appear in any of F or F's ancestor fragments, and (ii) X_i appears in G. *Remark:* A starting variable X_i in T_{FG} is not an active variable in F or F's ancestor fragments but becomes an active variable after entering G from F. The value of X_i is the same as the initial value when we enter G from F.

The set of coupling variables between F and G, C_{FG}: A variable X_i is in C_{FG} if and only if (i) X_i appears in one of F or F's ancestor fragments, and (ii) X_i appears in one of G or G's descendant fragments. The coupling variables in C_{FG} are active variables both in F and G. The coupling variables in C_{FG} are the media that propagates the effects of plan execution before F and in F to G.

Property 3 *For any coherent fragment F and a child coherent fragment G of F, the following equations are true: (i) $C_{FG} = A_F - T_{FG}$, (ii) $A_G = C_{FG} \cup S_{FG}$, (iii) $C_{FG} = A_G - S_{FG}$, and (iv) $A_F = C_{FG} \cup T_{FG}$.*
Remark: Consider the point in time when we leave fragment F to enter fragment G. We can determine the values of the active variables of G (F) from the values of the active variables of F (G) at that time by (i) and (ii) (by (iii) and (iv))

4. Exploit Locality in High-level Plans

In this section, we investigate the relationship between the optimal policy of a high-level plan and the Markov decision processes associated with the coherent fragments of the plan. We develop an algorithm to construct these Markov decision processes by dynamic programming. For each coherent fragment, we determine an optimal local policy by solving an associated Markov decision process. The union of these optimal local policies for coherent fragments then provides an optimal policy for the entire plan.

4.1. Constructing Markov Decision Processes over Coherent Fragments

In the following, we define the *global phase space* in executing a high-level plan. We then describe the *local phase spaces* of the coherent fragments, which introduce dimensionality reduction. Finally we demonstrate how to construct a Markov decision process over the local phase space of a coherent fragment F.

The global phase space: Given a high-level plan H, a global phase $(S, v_1, \ldots, v_n)$ specifies a situation during plan execution where (i) we are at a point in time immediately prior to taking an action enabled in a plan step S and (ii) $v_1, \ldots, v_n$ are the values of $X_1, \ldots, X_n$ respectively. The global phases space Phase(H) is composed of all possible global phases of when executing plan H.

Active state subspaces: For a coherent fragment F, $\text{SubSpace}(F) = \prod_{X_i \in A_F} \Omega_{X_i}$ is the active state subspace of F. An abstract state in SubSpace(F) specifies the values of the active variables of F. *Remark:* Rather than considering the entire state space, we only need to consider the active state subspace during the plan execution of a particular coherent fragments.
We use the variable $\bar{X}_F = (X_F^1, X_F^2, \ldots)$ to represent an *abstract* state in SubSpace(F), where $X_F^1, X_F^2, \ldots$ are the active variables in F.

Local phase spaces: For a coherent fragment F, $\text{Phase}(F) = F \times \text{SubSpace}(F)$ is the local phase space of F. A local phase in Phase(F) specifies (i) a plan step in F and (ii) the values of the active variables in F. *Remark:* Compared with the global phase space, Phase(F) ignores the redundant information about the passive variables during the plan execution in F. We use the variable $(S, \bar{X}_F)$ to represent a local phase in Phase(F), where S is a plan step in F and $\bar{X}_F$ is an abstract state in SubSpace(F).

The sources in a local phase space: For a coherent fragment F, Source(F) is composed of those phases $(S, \bar{X}_F)$ in Phase(F) where (i) S is a plan step in F, (ii) the guard expression G_S of plan step S is true with respect to the abstract state $\bar{X}_F$, and (iii) S has a parent plan step in one of F's parent fragments. *Remark:* Source(F) is composed of all of the possible local phases when we enter fragment F for the first time.

The targets of a local phase space: For a coherent fragment F, Target(F) is the union of Source(G) over every child coherent fragment G of F. *Remark:* Target(F) is composed of all the possible local phases when we leave F and enter one of F's child

fragment for the first time.

The Markov decision process associated with a coherent fragment:
For each coherent fragment F, we can construct a Markov decision process $\text{MDP}(F) = (\text{Phase}(F), \Omega_A, \text{Source}(F), \text{Target}(F), \tilde{\Pr}, \tilde{\text{Cost}})$ in the following way.

$\text{Phase}(F)$ is the space considered in $\text{MDP}(F)$. At the beginning, we are in one of the phases in $\text{Source}(F)$. Ω_A is the action space considered in $\text{MDP}(F)$. By taking actions, we move from phase to phase in $\text{Phase}(F)$ until we reach a target in $\text{Target}(F)$.

The conditional probabilities and the costs associated with actions and action outcomes in $\text{MDP}(F)$ are specified by $\tilde{\Pr}$ and $\tilde{\text{Cost}}$ respectively, which are slight modifications of Pr and Cost in the domain dynamics $\langle V, \text{Space}(V), \Omega_A, \Pr, \text{Cost}\rangle$ described in Section 2.1.

1. An action k can cause a transition from a phase $(S, \bar{X}_F)$ to a phase $(R, \bar{X}'_F)$ in $\text{Phase}(F)$ or a target $(R, \bar{X}_G)$ in $\text{Target}(F)$ *only if* (i) k is enabled in plan step S, (ii) plan step R is a child of S, and (iii) the guard expression G_R of R is true in $\bar{X}'_F$ (or $(R, \bar{X}_G)$).

2. In the case of transition to a phase $(R, \bar{X}'_F)$ in $\text{Phase}(F)$, the conditional probability and the cost of such a phase transition are the same as $\Pr(\bar{X}'_F|k, \bar{X}_F)$ and $\text{Cost}(\bar{X}'_F|k, \bar{X}_F)$ respectively.

3. In the case of transition to a target $(R, \bar{X}_G)$ in $\text{Phase}(F)$, the conditional probability of such a phase transition is the same as $\Pr(\bar{X}_G|k, \bar{X}_F)$. However, the cost $\tilde{\text{Cost}}(\bar{X}_G|k, \bar{X}_F)$ is modified as (i) the original $\text{Cost}(\bar{X}_G|k, \bar{X}_F)$ plus (ii) the expected cumulative cost of $(R, \bar{X}_G)$ in $\text{MDP}(G)$ minus (iii) $\sum_{X_i \in T_{FG}} r_i v_i$ where v_i is the value of X_i in $\bar{X}_F$.

4. Regarding the modification of action cost in 3, the first term specifies the actual action cost. The second term reflects the portion of final reward received from the terminal variable in T_{FG}, since they remain the same till the end as soon as we enter G. The third term reflects the cumulative cost minus the portion of final reward received from the variables whose values may change even after entering G.

Proposition 1 *We can determine an optimal policy* π_F *of* $\text{MDP}(F)$ *in time polynomial in the size of* $\text{Phase}(F)$. [6]

Proposition 2 π_F *maps each phase in* $\text{Phase}(F)$ *to an action and minimize (i) the expected cumulative cost minus (ii) the expected final reward since entering* F *until finishing the plan execution.*

4.2. Deriving Optimal Policies of Plans

The following algorithm exploit locality structure embedded in a high-level plan to expedite the derivation of an optimal policy.

[6] This can be achieved by applying standard techniques to solve Markov decision processes [7] [8].

Input: (i) the domain dynamics $\langle V, \text{Space}(V), \Omega_A, \Pr, \text{Cost}\rangle$, (ii) a high-level plan $H = (N_H, E_H)$, (iii) a function of final reward $\sum_{X_i \in V} r_i X_i$, and (iv) the initial values of the variables in V.

Output: An optimal policy for the plan.

1. Identify the coherent fragments and derive the fragment graph from the high-level plan H by finding strongly-connected components in H [1].
2. Identify (i) the sets of active variables for coherent fragments, (ii) the sets of terminal variables, starting variables, and coupling variables among coherent fragments by scanning through the plan steps in H.
3. Repeat Step 4 and Step 5 until all coherent fragments are processed.
4. Pick a coherent fragment F with no child coherent fragments or all of whose child coherent fragments have been processed. Construct the Markov decision process MDP(F) according to (i) the information in step 1 and step 2 and (ii) the information propagated from the child coherent fragments of F.
5. Derive the optimal local policy π_F for MDP(F). Evaluate the expected cumulative of each phase in Source(F) according to π_F, and propagate this information to the parent coherent fragments of F.
6. Report the union of π_F over all coherent fragments F's as the optimal policy for H.

5. Analysis and Discussion

A nice property of our algorithm is that the time complexity does not directly depend on the size of the state space Space(V). Note that $|\text{Space}(V)| = 2^{|V|}$. In general, it is beyond our capability to explore the underlying state space as a whole, since a reasonable domain normally involves a large number of state variables. By carefully exploiting locality embedded in a high-level plan our algorithm can greatly expedite the computation.

Given a high-level plan H, we define $m = \max_F |\text{Phase}(F)| = \max_F |F| \times 2^{|A_F|}$ where A_F is the set of active variables in coherent fragment F. Let poly(m) be the amount of time spent in deriving and evaluating an optimal policy of a Markov decision process of size m. (By Proposition 1, poly(m) is a polynomial function of m.) Let N be the total number of the plan steps in the plan H. Our algorithm has $O(N \times \text{poly}(m))$ time complexity. This is because we have at most N coherent fragments in H. For each coherent fragment, we solve the associated Markov decision process of size m.

Example: Figure 4 depicts the Source(F), Target(F) and the active variables in each coherent fragment F of the quality improvement plan. Since each coherent fragment has two plan steps and four active variables, an optimal policy can be determined by solving four Markov decision processes, each of whose local phase space is of size $2 \times 2^4 = 32$. This is much better than naive exploration of Space(V), which is of size $2^{10} = 1024$ since we have ten state variables.

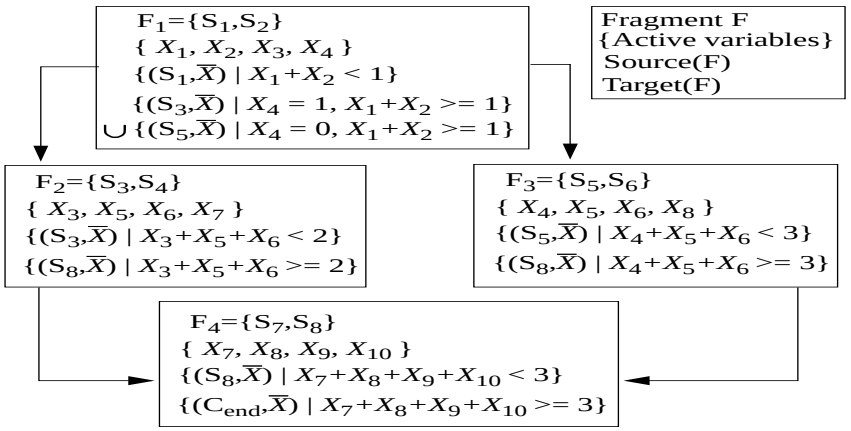

Figure 4: Information to construct associated Markov decision processes

Locality in a problem instance is measured by m, which is determined by the sizes of coherent fragments and the numbers of active variables involved in individual coherent fragments. Small m indicates that plan execution of H is highly localized.

1. If $m = O(1)$, our algorithm takes time linear in the total number of plan steps N. This happens when $|F| = O(1)$ and $|A_F| = O(1)$ for every coherent fragment F.
2. If $m = O(N^c)$ for any constant c, our algorithm takes time polynomial in the number of the total plan steps N. This happens when $|F| = O(N)$ and $|A_F| = O(\log N)$ for every coherent fragment F.
3. If $m = O(2^N)$, our algorithm degrades to take time exponential in N. This happens if $|A_F| = O(N)$ for a coherent fragment F.

The first case stands for problem instances that can be solved very efficiently. The third case is rare and hopelessly complicated, where a very large number of state variables are involved at a time in a fragment. Since the size of any coherent fragment F is no more than N, we have the following proposition according to the second case.

Proposition 3 *We can derive an optimal policy in time polynomial in the number of total plan steps N given that each coherent fragment in the input high-level plan has $O(\log N)$ active variables.*

6. Related Work

The particular representation of Markov processes in terms of high-level plans with conditional branches and loops that serves as inspiration for this paper is due to Smith and Williamson [10]. The particular methods for compiling fragment graphs are adapted from the work of Lin and Dean [6] on expediting temporal inference, which in turn builds on the work of Lansky [5], Tenenberg [11] and others in exploiting locality planning.

The general idea of restricting reachability in state space by enabling some actions and disabling others is common in control theory. See Ramadge and Wonhan [9] for

an survey of work on discrete event systems that concerns the synthesis of supervisory systems. Dean and Lin [3] describe a family of techniques for decomposing the computations required to solve large Markov decision processes. This family of techniques complements the representations and algorithms investigated in this paper.

References

[1] Corman, T. H., Leiserson, C. E., and Rivest, R. L., *Algorithms*, (MIT Press, Cambridge, Massachusetts, 1991).

[2] Dean, Thomas and Kanazawa, Keiji, A Model for Reasoning About Persistence and Causation, *Computational Intelligence*, **5**(3) (1989) 142–150.

[3] Dean, Thomas and Lin, Shieu-Hong, *Decomposition Techniques for Planning in Stochastic Domains*, Technical Report CS-95-, Brown University Department of Computer Science, 1995.

[4] Hanks, Steve and McDermott, Drew V., Modeling a Dynamic and Uncertain World I: Symbolic and Probabilistic Reasoning About Change, *Artificial Intelligence*, (1994).

[5] Lansky, Amy L., Localized Event-Based Reasoning for Multiagent Domains, *Computational Intelligence*, **4**(4) (1988).

[6] Lin, Shieu-Hong and Dean, Thomas, Exploiting Locality in Temporal Reasoning, Sandewall, E. and Backstrom, C., (Eds.), *Current Trends in AI Planning*, Amsterdam, IOS Press, 1994.

[7] Papadimitriou, Christos H. and Tsitsiklis, John N., The Complexity of Markov Chain Decision Processes, *Mathematics of Operations Research*, **12**(3) (1987) 441–450.

[8] Puterman, Martin L., *Markov Decision Processes*, (John Wiley & Sons, New York, 1994).

[9] Ramadge, Peter and Wonham, Murray, The Control of Discrete Event Systems, *Proceedings of the IEEE*, **77**(1) (1989) 81–98.

[10] Smith, David E. and Williamson, Mike, Representation and Evaluation of Plans with Loops, To appear in the working notes for the 1994 Stanford Spring Symposium on Extended Theories of Action, 1995.

[11] Tenenberg, Josh D., Abstraction in Planning, Allen, J. F., Kautz, H. A., Pelavin, R. N., and Tenenberg, J. D., (Eds.), *Reasoning about Plans*, (Morgan Kaufmann, San Francisco, California, 1991), 213–280.