Title:

Goal Management in Organizations: A Markov Decision Process (MDP) Approach[*]

(*Student Paper Submission*)

Suggested Track:

Modeling & Simulation

Authors:

Candra Meirina
Graduate Student, ECE Dept., UCONN
Storrs, CT
Fax: 860-486-5585
Phone: 860-486-2210
e-mail: meirina@engr.uconn.edu

Yuri N. Levchuk
Aptima, Inc.
Woburn, MA
Phone: 781-935-3966x236
e-mail: levchuk@aptima.com

Georgiy M. Levchuk
Graduate Student, ECE Dept., UCONN
Storrs, CT
Fax: 860-486-5585
Phone: 860-486-2210
e-mail: georgiy@engr.uconn.edu

Krishna R. Pattipati[1]
Professor, ECE Dept., UCONN
Storrs, CT
Fax: 860-486-5585
Phone: 860-486-2890
e-mail: krishna@sol.uconn.edu, krishna@engr.uconn.edu

David L. Kleinman
Professor, Naval Postgraduate School
589 Dyer Road, Room 200A
Monterrey, CA
Fax: 831-656-3679
Phone: 831-656-4148
e-mail: kleinman@nps.navy.mil

[*] This work was supported by the Office of Naval Research under contract # N00014-00-1-0101.
[1] The author to whom correspondence should be addressed.

Goal Management in Organizations: A Markov Decision Process (MDP) Approach[*]

Candra Meirina
Dept. of Electrical and Computer Engineering
The University of Connecticut
260 Glenbrook Road, Storrs, CT 06269-2157
Phone: (860)-486-2210, E-mail: meirina@engr.uconn.edu

Yuri N. Levchuk
Aptima, Inc.
Woburn, MA
Phone: 781-935-3966x236, E-mail: levchuk@aptima.com

Georgiy Levchuk
Dept. of Electrical and Computer Engineering
The University of Connecticut
260 Glenbrook Road, Storrs, CT 06269-2157
Phone: (860)-486-2210, E-mail: georgiy@engr.uconn.edu

Krishna R. Pattipati[2]
Dept. of Electrical and Computer Engineering
The University of Connecticut
260 Glenbrook Road, Storrs, CT 06269-2157
Phone: (860)-486-2890, Fax (860)-486-5585, E-mail: krishna@sol.uconn.edu

David L. Kleinman
Naval Postgraduate School
589 Dyer Road, Room 200A, Monterrey, CA
Fax: 831-656-3679, Phone: 831-656-4148, E-mail: kleinman@nps.navy.mil

[*] This work was supported by the Office of Naval Research under contract # N00014-00-1-0101.
[2] The author to whom correspondence should be addressed.

Abstract

Goal management is the process of recognizing or inferring goals of individual team members; abandoning goals that are no longer relevant; identifying and resolving conflicts among goals; and prioritizing goals consistently for optimal team collaboration and effective operations. A Markov decision process (MDP) approach is employed to maximize the probability of achieving the primary goals (a subset of all goals). We seek to address the computational adequacy of an MDP as a planning model by introducing novel problem domain-specific heuristic evaluation functions (HEF) to aid the search process. We employ the optimal AO search and two suboptimal greedy search algorithms to solve the MDP problem. A comparison of these algorithms to the dynamic programming algorithm shows that computational complexity can be reduced substantially. In addition, we recognize that embedded in the MDP solution, there are a number of different action sequences by which a team's goals can be realized. That is, in achieving the aforementioned optimality criterion, we identify alternate sequences for accomplishing the primary goals.*

1. Introduction

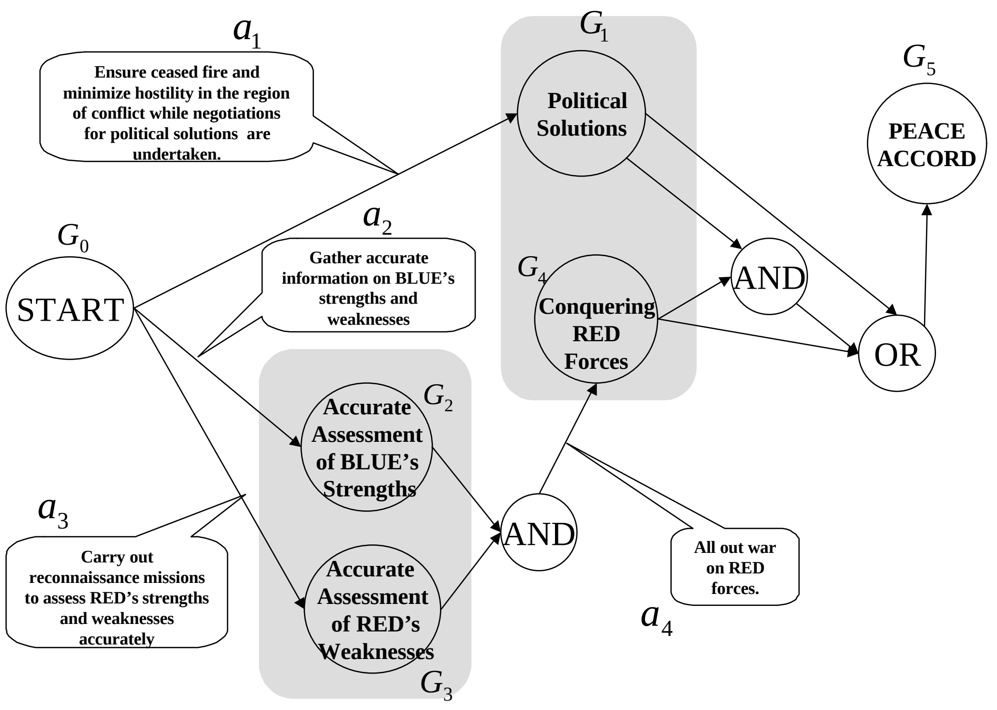
1.1. *Motivation*

Changing patterns of today's world impose the need for current and future military forces to conduct a broader and more complex spectrum of operations. Wars no longer take place between nation-states on traditional battlefields, but have been replaced by emergent and asymmetric threats involving cultural factions and trans-national players. Decisions must be made in real time with simultaneous tactical, operational, and strategic implications. In response to these demanding requirements, military forces need to employ new operational concepts and command approaches. This calls for much greater emphasis on realistic modeling of dynamic military organizations that enable them to evaluate their current strategies, their strengths and weaknesses, and explore various strategic options based on current knowledge and forecasts.

A team management mechanism, which ensures the cooperation of individuals in their pursuit of desired organizational goals, involves both managing the team's intentions and organizing its activities to fulfill them. A desired *system state* describes organization's internal and external conditions. It is comprised of many independent or loosely dependent dimensions of the system and its environment (i.e., set of desired goals, resources available to pursue the goals, time available for mission processing, etc.). Deliberate changes in states are brought out by *functions* or (a set of) *actions*, which are assigned to individuals within a team. That is, a *function* can imply a *specific intent to change the state of the environment* or *an activity carried out by an individual(s) to perform this change*. *Goal management* is the process of *prioritizing* goals consistently for optimized team collaboration. The goal management problem is formulated as a Markov decision problem, in which the objective is to determine an optimal closed-loop policy that maximizes the probability of reaching the final desired system state under resource and time constraints.

In order to elucidate the points made throughout the paper, let us consider the following simplified scenario. Assume that there are two opposing coalitions: the RED and the BLUE factions. *Political factors* (introduced by broad coalitions of all parties involved), *the geographic spread* of BLUE military industrial complex (the location of maintenance facilities, army depots and logistics pipelines), *mixed ethnicities, cultures, and religious backgrounds* among the parties involved, significantly affect the course of operations. This situation produces complications with a direct bearing on the course of military activities. With enormous humanitarian, political, and economic stakes, the BLUE forces resolve to bring peace to all factions involved.

BLUE's political and military experts around the globe agree that there are three major avenues to resolving the conflict. The first is a pure political solution, wherein all parties involved would resolve their differences through negotiations and constructive talks. In this case, the involvement of military forces would be restricted to minimizing hostilities in the region of conflict. The second solution proposes an all out war to remove RED's forces from the region. To ensure a successful outcome, this approach will require BLUE to have accurate assessments on the strengths and weaknesses of both sides. In order to do this, reconnaissance missions need to be conducted to accurately estimate RED's forces. Based on the gathered information, strategies for an all-out war are drawn. The BLUE's experts also suggest combining the former strategies to produce a comprehensive solution involving both political and military approaches. Suppose the BLUE's military strategists are weighing these three approaches and are analyzing the possible outcomes. The tactical roadmap of options and outcomes can be described by an AND/OR graph as in Figure 1. The task is to identify a strategy, which maximizes the possibility of reaching a peace accord, given specified resource and time constraints.



a_1
Ensure ceased fire and minimize hostility in the region of conflict while negotiations for political solutions are undertaken.
G_1
Political Solutions
G_5
PEACE ACCORD
a_2
Gather accurate information on BLUE's strengths and weaknesses
G_0
START
G_4
Conquering RED Forces
AND
OR
G_2
Accurate Assessment of BLUE's Strengths
a_3
Carry out reconnaissance missions to assess RED's strengths and weaknesses accurately
AND
All out war on RED forces.
a_4
Accurate Assessment of RED's Weaknesses
G_3

Figure 1. The AND/OR Goal Graph of BLUE Coalition

1.2. *Representation of the Problem*

The Markov decision process-based strategy roadmap connecting the initial system state to the final state, via a set of intermediate states, is represented by an acyclic graph. The *nodes* of the graph denote *system states*. The *arcs* denote *transition probabilities* among system states, which depend on functions executed at the state and on the amount of resource and time available at this state. Each function requires a certain amount of resource to complete[3]. Transition probabilities to move from one state to another, given the control function applied at a state, are given. The objective is to find a sequence of control functions (actions) that maximize the probability of reaching the final goal state under resource and time constraints.

More formally, the approach can be formulated as follows. Let $S = [X, R, D]$ denote a state space that includes the *resource state space* R and the duration dimension D (such that $D = [0, d], d \in I$ (integer), and $R = [0, r], r \in I$). The notation $X = \{1, 2, \Lambda\ , n\}$ [4] denotes the *state space* whose elements represent different '*states* of the *system goals*'. Formally, the system state is characterized by the three-tuple $\underline{s} = (i, \underline{y})$, where i denotes the state of goals, $\underline{y} = (r, d)$ with r denoting the available resources and d denoting the available time.

Let $F(i, \underline{y}) = \{ f_m(i, \underline{y}), m = 1, ..., M(i, \underline{y})\}$ be a set of control functions that can be applied when the system state is $\underline{s} = (i, \underline{y})$. The notation $d(f_m(i, \underline{y})) \geq 0$ defines the *duration* of applying a control function $f_m(i, \underline{y})$ in state i, while $r(f_m(i, \underline{y})) \geq 0$ denotes *resource requirements* for control function $f_m(i, \underline{y})$. A function $f_m(i, \underline{y})$ transitions the goal state $i \in X_i$ to the goal state $j \in X_j$ in $d(f_m)$ time units, while utilizing $r(f_m)$ resources.

f_i	a_4	a_3	a_2	a_1	$r(f_m)$	$d(f_m)$
f_1	0	0	0	1	3	4
f_2	0	0	1	0	1	1
f_3	0	1	0	0	2	2
f_4	1	0	0	0	3	2
f_5	0	0	1	1	4	4
f_6	0	1	0	1	5	4
f_7	0	1	1	0	3	2
f_8	0	1	1	1	6	5
f_9	1	1	0	0	5	3

Table I. Functions Available to BLUE Forces to Achieve Intended Goals

Let the *strategy roadmap* denote a directed graph $\Omega(S, V)$ consisting of *primary nodes* $S = \{\underline{s}\}$ representing system states and *edges* $V = \{v\}$. The edges denote transition probabilities (i.e., probabilities of reaching the intended system states when certain *control functions* are carried out). Let $p_{ij}(f_m(i, \underline{y})) = P(x_{k+1} = j \mid x_k = i, \underline{y}, f_m(i, \underline{y}))$ denote the transition probability that the goal state at stage $k+1$ is j given that the goal state at stage k is i and that a control function $f_m(i, \underline{y})$ is applied. When function $f_m(i, \underline{y})$ is executed by the system, this execution takes $d(f_m(i, \underline{y}))$ units of time and expends $r(f_m(i, \underline{y}))$ resource units. Therefore, when function $f_m(\underline{s}_k)$ is applied at state $\underline{s}_k = (i, \underline{y}_k)$,

[3] For elucidation purposes, we consider only one resource type. Extension to vector of resources is straightforward.

[4] The notation $x_k = i$ means that the system goal state is i at stage k.

the system state probabilistically changes to $\underline{s}_{k+1} = (j, \underline{y}_{k+1})$, where $\underline{y}_{k+1} = [r_k - r(f_m(i, \underline{y}_k)), d_k - d(f_m(i, \underline{y}_k))]$. The probabilistic notions are introduced to accommodate various unforeseen events, such as execution failures and random state shifts during the execution of control functions. Based on the problem constraints, we generate a layered acyclic graph of system states with N layers. The objective is to maximize the probability of reaching the final goal states $P(x_N = l, l \in X_N)$, subject to $\underline{y}_0 = (r, d)$, $\underline{y}_k \geq \underline{0}, \forall k = 1, \ldots, N$.

x_k	G_5	G_4	G_3	G_2	G_1	G_0	$F(x_k)$	x_{k+1}
1	0	0	0	0	0	1	$f_1, f_2, f_3, f_5, f_6, f_7, f_8$	1,2,3,4,5,6,7,8, [11,12,13,14]
2	0	0	0	0	1	1	$f_2, f_3, f_5, f_6, f_7, f_8$	2,4,6,8, [11,12,13,14]
3	0	0	0	1	0	1	$f_1, f_3, f_5, f_6, f_7, f_8$	3,4,7,8, [12,14]
4	0	0	0	1	1	1	f_3, f_6, f_7, f_8	4,8, [12,14,16]
5	0	0	1	0	0	1	$f_1, f_2, f_5, f_6, f_7, f_8$	5,6,7,8, [11,12,13,14]
6	0	0	1	0	1	1	f_2, f_5, f_7, f_8	6,8, [13,14]
7	0	0	1	1	0	1	$f_1, f_4, f_5, f_6, f_8, f_9$	7,8,9,14
8	0	0	1	1	1	1	f_4, f_9	8,10, [14,16]
9	0	1	1	1	0	1	f_1, f_5, f_6, f_8	9,10, [15,16]
10	0	1	1	1	1	1	f_9	10,16
11	1	0	0	0	1	1	-	Terminal (Absorbing) State
12	1	0	0	1	1	1	-	
13	1	0	1	0	1	1	-	
14	1	0	1	1	1	1	-	
15	1	1	1	1	0	1	-	
16	1	1	1	1	1	1	-	

Table II. Markov State Representation of AND/OR Goal Graph

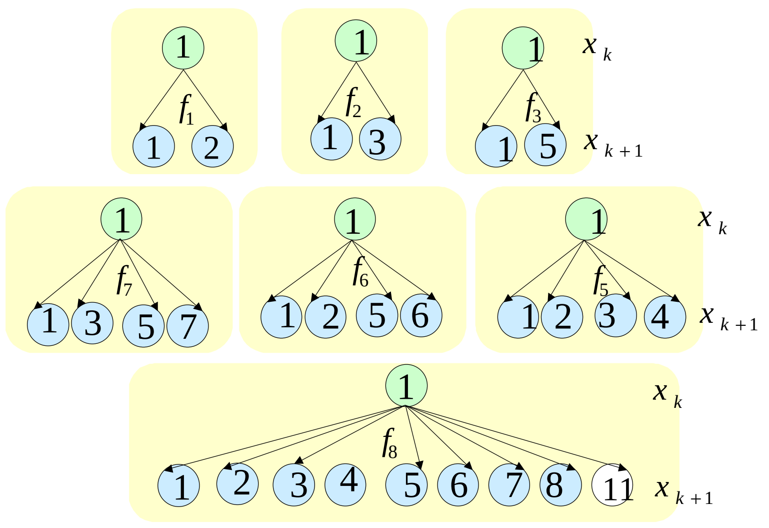


Figure 2. The Transitions from $x_k = 1$ to Successors $x_{k+1} = j$ Given $f_m(1, \underline{y})$

Let us examine the scenario presented in Figure 1. In Table II, we can see the transformation of AND/OR graph nodes in Figure 1 into the X states of a strategy roadmap. The nodes no longer denote the goal states exclusively; they become representations of the system states. For example, $x_k = 6$ represents the system state in which goals G_1 and G_3 have been achieved (G_0 represents the known initial system state). Furthermore, if the system is in state $x_k = 6$, it can transition to $x_{k+1} = 6,8,13$, or 14 by applying f_2, f_5, f_7, or f_8. That is, the strategy roadmap representation naturally captures the options and restrictions of the system at each state. If the system is in $x_k = 6$, the function f_9 may not be executed, for this requires G_2 to have been achieved. Moreover, being in $x_k = 6$ would make it impossible for the system (given its allowable options) to reach $x_{k+1} = 15$. In addition, being in state $x_k = 6$ also makes state 5 infeasible. That is, after reaching G_1, it is not viable for the system to abandon it. Although this assumption is somewhat restrictive, it is a good assumption in many situations. For example, it is a good assumption if the goal is to destroy a target; if the goal is achieved, whatever the system does will not undo it.

In general, execution failures and random state shifts during the execution of control functions may actually transform the system to states other than the intended one. The MDP based strategy roadmap representation easily accommodates this case. For example, assume that the system is at $[6, r, d]$. When function f_8 is applied (that is assuming that $r - r(f_8) \geq 0$ and $d - d(f_8) \geq 0$), due to various internal and external events, the system may transition to $[6, r - r(f_8), d - d(f_8)]$, $[8, r - r(f_8), d - d(f_8)]$, $[13, r - r(f_8), d - d(f_8)]$, or $[14, r - r(f_8), d - d(f_8)]$ (See Table IV in the Appendix). Furthermore, in order to reduce the number of system states, we can combine all the absorbing states into one. That is, noting that $x_k = 11 = 12 = 13 = 14 = 15 = 16$ all represent the terminal states, we can simply label them as $x_k = 11$.

1.3. *Contributions and Earlier Work*

The goal management problem belongs to the class of planning problems under uncertainty (which is typically termed decision-theoretic planning (DTP)). We adopt a Markov decision process (MDP) framework as an underlying model for the problem.

Formulation of planning problems, and in particular probabilistic planning problems, as graph search problems (specifically as MDP) has attained a surge of interest in recent years, [11], [12]. See also [5], [6], [13], and [15],. An MDP is a suitable representation of goal management problems, since the model takes into consideration problem uncertainties, which include uncertain effects of actions, incomplete information about the environment, and uncertainties in the goal states. Moreover, it models our problem accurately, since the objective of an MDP is to devise courses of actions (plans or policies) with a high probability of success, in contrast to an assured attainment of intended goals as in a traditional deterministic planning problem.

Although the representation of a DTP as an MDP is not new, the explicit connection between the traditional planning routines (in particular AND/OR graph representations, which exclude uncertainties) and the MDP is novel. Adopting the MDP framework as a model for formulating and solving goal management problems has its advantages and shortcomings. Goal management problems typically exhibit considerable structure in value functions describing the performance criteria, in functions depicting state transitions, and in relationships among features used to describe states and actions. These permit the use of special purpose methods that recognize and exploit that structure; thereby allowing it to be solved with less computational effort than other methods. Specifically, the MDP representation takes into account the problem domain specifics and uses them to its advantage, viz., optimizing the cardinality of the state space. In the given scenario, the cardinality of the system states $|S|=11$ instead of 2^5 (the example encompasses 5 goals); the cardinalities of the control function sets $|F|$ are between 1 to 7, instead of 2^4 (the example has 4 possible actions). See Table I and Table II.

The general impediment [6] to the more widespread acceptance of MDPs as a general model of planning is the computational adequacy of MDPs as a planning model: can the techniques scale to solve planning problems of reasonable size? One difficulty with the solution techniques for MDPs is the tendency to rely on explicit, state-based problem formulations. This can be problematic, since state space grows exponentially with the number of problem features (e.g., multiple types of resources, large number of control actions, etc.). This paper seeks to alleviate the computational burden in the MDP-based approach by introducing problem domain-based heuristics into the search algorithms.

1.4. *Organization of the Paper*

We formulate the goal management problem as a finite-state Markov decision problem. This approach opens doors to utilizing a broad field of mature graph search techniques. In particular, we explore various optimal and sub-optimal heuristic search approaches and contrast the solutions (for small problems) to those of dynamic programming (DP) recursions.

The paper is organized as follows. Section 2 explores various graph search techniques, which include the classical DP techniques, AO* heuristic search algorithm, and greedy heuristics. Novel problem domain-based heuristic evaluation functions (HEFs) are introduced and evidence of their admissibility is presented. Furthermore, greedy heuristic search techniques are also considered in this paper. In section 3, we compare various algorithms on an illustrative example. Finally, we conclude with a summary of findings and future work.

2. Solution Approaches

The paper seeks to alleviate the computational burden inherited by MDP-based approaches through the use of problem domain-based heuristics. In particular, the paper explores the optimal AO* algorithm, and several suboptimal greedy heuristics. The (computational) costs and performances of the aforementioned approaches are then contrasted with the standard DP recursions.

2.1. *Dynamic Programming (DP) Recursion*

The DP is a common technique used in situations when decisions can be made in stages and when, despite the unpredictable nature of the decision outcomes, the desired outcome can be quantified in advance. The idea is to quantify the desired outcome in a mathematical expression (typically referred to as the objective function) and the goal is to maximize (minimize, depending on the nature of the problem) it. An important aspect of the problem is that a decision, made at each stage, is made based on the present value function and expected future value [1] and [2]. We are interested in knowing which set of control functions $\{f_m(i,\underline{y})\}$ to apply at each stage such that it results in maximal probability of reaching the final goal.

We represent the next goal state $x_{k+1} = j$ as a function of the current goal state $x_k = i$ and the control function being applied $f_m(i, \underline{y})$:

$$j = g(i, f_m(i, \underline{y})) \qquad (1)$$

Furthermore, we consider control laws that consist of a sequence of admissible functions $p = \{f^{(0)}, f^{(1)}, ..., f^{(N-1)}\}$[5], defined by $f^{(k)} = \{m_k(s_k), k = 0, \ldots, N-1\}$. The notation $m_k(\bullet)$ signifies some function of the argument. Given an initial state $s_0 = (x_0, \underline{y}_0)$ and an admissible policy $p = \{f^{(0)}, f^{(1)}, ..., f^{(N-1)}\}$, the objective function is given by

$$J_p(x_0, \underline{y}_0) = P(x_N = l, l \in X_N) \qquad (2)$$

where X_N denotes the set of absorbing states (intended goal states). Recall that when the absorbing states are lumped into one state, we set $X_N = n$ so that $|X_N| = 1$. Thus, an optimal policy p^* is one, which maximizes the following objective function:

$$J_{p^*}(x_0, \underline{y}_0) = \max_p J_p(x_0, \underline{y}_0) \qquad (3)$$

The DP technique is based on the principle of optimality. See [1] and [2]. In words, it states that if there exists an optimal policy $p_0^* = \left\{f^{(0)*}, f^{(1)*}, ..., f^{(N-1)*}\right\}$, then the truncated policy $p_k^* = \left\{f^{(k)*}, f^{(k+1)*}, ..., f^{(N-1)*}\right\}$ is also optimal. The DP decomposes the problem into a sequence of optimizations carried over the set of control functions. The DP algorithm starts with $J_N(x_N, \underline{y}_N)$ and proceeds backward in stages from stage $N-1$ to 0. We let $J_k(i, r, d)$ be the *cost-to-go* (value-to-go in our case) function at stage k. From the principle of optimality, the dynamic programming (DP) algorithm can be written as

$$J_k^*(i, r, d) = \max_{\substack{f \in F(i, \underline{y}) \\ r(f) \le r, d(f) \le d}} \left[\sum_j p_{ij} J_{k+1}^*(j, r - r(f), d - d(f)) \right], k = N-1, \ldots, 0 \qquad (4)$$

with a terminal condition

$$J_N^*(n, r, d) = 1,\ J_N^*(i, r, d) = 0, \forall i \ne n, r \ge 0, d \ge 0 \qquad (5)$$

However, the DP algorithm has computational requirements of $O((|X|M)^Q)$, where $M = \max\{|F|\}$ denotes the largest cardinality of the control function sets, $|X|$ represents the cardinality of the goal states, and $Q = \min\{r/\max_f r(f), d/\max_f d(f)\}$. This can be quite substantial for large $M, |X|$, and r (or d).

[5] Note that superscripted $f^{(k)} = \{f_m(\underline{s}) : f_m(\underline{s}) \in F(\underline{s}), \underline{s} \in S_k\}$ denotes a set of control functions applied at the k-th stage, S_k signifies the set of system states at stage k; whereas subscripted f_m signifies the m-th control function.

2.2. *Heuristic Approaches*

Another approach for planning under uncertainty is based on state-based graph search [6]. These techniques employ estimates of value-to-go functions in (4), called heuristic evaluation functions (HEFs), to overcome the computational explosion of the DP recursion. In the following, we consider two such HEFs, as well as greedy heuristics that employ local step-by-step optimization.

2.2.1. *Heuristic Evaluation Functions (HEFs)*

We formulate the problem of maximizing the conditional probability of reaching a final goal state under resource and time constraints as an informed best-first search on an MDP-based graph, wherein the approximate value-to-go (HEF) is derived from the current conditional probability values of a partially developed tree and the expected depth of the tree (the residual layers, based on the remaining resources and time). The HEF allows for the use of a top-down search algorithm, such as AO* [16]. In the following, we develop two HEFs.

2.2.1.1. *HEF 1:* $h_1(\underline{s})$

The first HEF follows directly from the DP recursion (4), and the fact that the optimal value-to-go is bounded by 1:

$$J_k^*(i,r,d) \le \max_{\substack{f\in F(i,\underline{y}) \\ r(f)\le r, d(f)\le d}} \sum_j p_{ij}(f), k=0,\Lambda\ , N-1 \qquad (6)$$

The upper bound depends on the value of the transition probability $p_{ij}(f_m(i,\underline{y})) = P(x_{k+1}=j \mid x_k=i, \underline{y}, f_m(i,\underline{y}))$. This bound determines our first HEF:

$$h_1(\underline{s}) = \max_{\substack{f\in F(i) \\ r(f)\le r, d(f)\le d}} \sum_j p_{ij}(f), \forall x_k = i, \forall k=0,\dots,N-1 \qquad (7)$$

2.2.1.2. *HEF 2:* $h_2(\underline{s})$

In addition to p_{ij}, $J_k^*(i,r,d)$ depends on the number of the remaining layers, $N_{\min}$ at stage k (before the system exhausts either r or d). Recall that the problem constraints restrict the number of stages, N, in the MDP graph. The minimum number of remaining stages is determined as follow:

$$N_{\min} = \min\left(\max_{f\in F(i)} \text{ceil}\left(\frac{r_{\min}}{r(f)}\right), \max_{f\in F(i)} \text{ceil}\left(\frac{d_{\min}}{d(f)}\right)\right) \qquad (8)$$

The minimum resource requirement $r_{\min}$ is the smallest amount of resources the system needs to expend to reach the desired state from the current state; whereas $d_{\min}$ is the shortest time required for the system to reach the desired state from the current state. Note that the operator $ceil(\cdot)$ rounds the number up to the next integer value. Using (8), we define the second HEF:

$$h_2(\underline{s}) = (\max_{\substack{f \in F(i) \\ r(f) \le r, d(f) \le d}} \sum_j p_{ij}(f))^{N_{\min}} \tag{9}$$

2.2.2. *Admissibility of the HEFs*

In this subsection, we seek to prove that the selected HEFs have the property of *admissibility* [16], [17]. When the HEFs are admissible, the objective function *approaches the optimal objective function value from above (in the case of maximization).*

Definition 1: Let $\Omega(S)$ be an MDP based strategy graph. An HEF $h(\underline{s})$ defined on $\Omega(S)$ is admissible if for each node $\underline{s} \in S$ in $\Omega(S)$, $h^*(\underline{s}) \le h(\underline{s})$, the optimal value-to-go. Moreover, this $h(\underline{s})$ is always finite with an upper bound of *1*.

The admissibility of the first HEF follows directly from (4).

$$h^*(s) = J_k{}^*(i,r,d) \le \max_{\substack{f \in F(i,\underline{y}) \\ r(f) \le r, d(f) \le d}} \sum_j p_{ij}(f) = h_1(s), k = 0, \Lambda \;, N-1 \tag{10}$$

In the same vein, the analysis for the admissibility of second HEF follows from:

$$\begin{aligned} h^*(\underline{s}) &= J_k^*(i,r,d) \\ &\overset{?}{\le} (\max_{\substack{f_k \in F(\underline{s}_k) \\ r(f) \le r, d(f) \le d}} \sum_j P(x_{k+1} \mid x_k, f_k))^{N_{\min}} = h_2(\underline{s}), \forall x_k = i, \forall k = 0,\dots,N-1 \end{aligned} \tag{11}$$

Unfortunately, the second HEF does not always result in an admissible heuristic. When $\exists x_{k+1} \ne j : \sum_{\underline{s}_{k+1} \in \Pi(\underline{s}_k)} \max_{f_k \in F(\underline{s}_k)} P(x_{k+1} \mid x_k, f_k) >> \max_{f_k \in F(\underline{s}_k)} \sum_j P(j \mid x_k, f_k)$ and $N_{\min} \approx N-k$, the second HEF is inadmissible. Additionally, when the cardinality of $|X|$ is large, which means that the value of $\max_{f \in F(i)} p_{ij}(f)$ is small (recall that $\sum_{j=1}^n p_{ij}(f) = 1$), the second HEF is also potentially inadmissible. Otherwise, the second HEF is in general admissible.

2.2.3. *AO* Algorithm*

AO* is a best-first search algorithm [16], [17], which expands only nodes with the most promising chance of reaching the goal nodes on the basis of the HEF. The algorithm utilizes three steps repeatedly. First, a top-down graph traversing operation follows the best current path and accumulates the set of nodes that are on the path and not yet expanded. Second, the procedure selects an unexpanded node and expands it. The HEF of all the successors of the expanded node are computed and adds these nodes to the graph. Third, a bottom-up value revising operation changes the HEF of the expanded node and propagates this change back to the initial node according to the DP recursion in (4).

Following [16], there are two important attributes of this heuristic search strategy applicable here. First, if the HEF is admissible, AO* guarantees an optimal solution. Second, the search efficiency depends critically on the degree to which the HEF approximates the optimal value-to-go.

In this paper, we use the following conventions. We label nodes that are expanded as $CLOSED$. Nodes that are generated, but not yet expanded, are termed $OPEN$. These two sets are maintained throughout the search process.

Algorithm AO*:

Step 1: Initially let the search graph $\Omega(S)$ consist of the start node (that is, the initial goal, the initial available resources, and allowable duration) $\underline{s}_0 = (x_0, \underline{y}_0)$, $\underline{y}_0 = (r, d)$. Set $G(\underline{s}_0) = h(\underline{s}_0)$. If $\underline{s}_0$ is a terminal node, then label $\underline{s}_0$ $CLOSED$ and exit with the solution; otherwise, label it $OPEN$.

Step 2: Repeat the following steps until $\underline{s}_0$ is labeled closed. Then exit with $J = G(\underline{s}_0)$ as the expected value and the marked solution tree as the function strategy.

- *Step2.1:* Compute a partial solution graph $\tilde{\Omega}$ in $\Omega(S)$ by tracing down the marked arcs in $\Omega(S)$ from the root node $\underline{s}_0$. Select for expansion a node $\underline{s}_i$ of $\tilde{\Omega}$ that has the smallest $h(\underline{s}_i)$ (initially $\underline{s}_i = \underline{s}_0$).
- *Step2.2:* Generate in $\Omega(S)$ all successors of $\underline{s}_i = [i, r, d]$ spanned by the allowable functions $f \in F(\underline{s}_i)$: $\underline{s}_j = [j, r - r(f), d - d(f)]$. Label all $\underline{s}_j$ as $OPEN$. For each immediate successor of $\underline{s}_i$ not already present in $\Omega(S)$, set $G(\underline{s}_j) = h(\underline{s}_j)$. If any $\underline{s}_j$ is a terminal leaf node, label $\underline{s}_j$ $CLOSED$.
- *Step2.3:* Create a temporary set Z of nodes consisting only of nodes $\underline{s}_i$.
- *Step2.4:* Repeat the following steps until Z is empty
  - *Step 2.4.1:* Remove from Z a node $\underline{s}_j$ such that no successor of $\underline{s}_j$ in $\tilde{\Omega}$ occurs in Z.
  - *Step 2.4.2:* Revise the value HEF of $\underline{s}_i$ as follows: $e = \max_{f_m \in F(\underline{s}_i)} \{\sum_j p_{ij} G(\underline{s}_j)\}$. Let k be the index of the function for which maximum occurs. Resolve ties arbitrarily, but give preference to $CLOSED$ nodes. Mark all arcs spanned by f_k (both in Z and $\Omega(S)$). Label $\underline{s}_i$ $CLOSED$ if all of its successors $\underline{s}_j$ are labeled $CLOSED$.
  - *Step 2.4.3:* IF $G(\underline{s}_i) \neq e$, THEN Set $G(\underline{s}_i) = e$.
  - *Step 2.4.4:* If $G(\underline{s}_i)$ changes its value in step 2.4.3 or if $\underline{s}_i$ is labeled $CLOSED$, then add to Z $\underline{s}_i$ and all of the ancestors of $\underline{s}_i$ along the marked path. Ignore ancestors of $\underline{s}_i$ not connected to $\underline{s}_i$ by marked arcs.

2.2.4. *Greedy Heuristics*

The term greedy heuristic refers to the notion that all of the approximation techniques in this category employ a local, step–by–step optimization. The optimization techniques are typically in the form of a *k*-step look-ahead procedure. In this subsection, we consider two such techniques, with $k = 1$.

2.2.4.1. *Greedy Heuristic 1 (GH1)*

To select the best control function for a given state, the system needs to estimate the probability of reaching the intended goal state n by executing each available function. Let $X_{k+1}(f)$ be the set of all direct successors of the current goal state x_k, transformed by control function $f \in F(x_k)$, and let $|X_{k+1}(f)|$ be the cardinality of the direct successor space. In our example, if the current state is $x_k = 1$, then $|X_{k+1}(f_1)| = 2$, $|X_{k+1}(f_5)| = 4$, and so on. The first greedy heuristic (GH1) works on the premise that a function with a larger $|X_{k+1}(f)|$ has a better chance of bringing the system to $x_N = n$ faster. If $U(f)$ is the heuristic value for executing control function $f \in F(x_k)$, then the next control function f^k is selected to maximize $U(f)$:

$$f^k = \arg \max_{f \in F(x_k)} \left(U(f)\right)$$
$$U(f) = \begin{cases} \sum_{f \in F(x_k)} |X_{k+1}(f : \underline{y}_{k+1} \geq 0)| & \text{if } \exists\, x_{k+1} \in X_N \\ |X_{k+1}(f : \underline{y}_{k+1} \geq 0)| & \text{else} \end{cases} \qquad (12)$$

2.2.4.2. *Greedy Heuristic 2 (GH2)*

In general, greedy search GH1 tends to seize immediate reward at the expense of long-term gain. To alleviate this propensity, we consider a heuristic that estimates the value of each control function, accounting for future values. In this vein, at each state, the next control function f^k is chosen to maximize

$$f^k = \arg \max_{\substack{f \in F(i) \\ r(f) \leq r, d(f) \leq d}} \left\{ \sum_j p_{ij}(x_{k+1} = j \mid x_k = i, f) \right\} \qquad (13)$$

The latter heuristic is the heuristic evaluation function discussed earlier. GH2 has computational complexity of $O(M\,|X|\,Q)$, where M, $|X|$, and Q are as previously defined. The computational reduction compared to that of DP, $O((|X|\,M)^Q)$, is substantial and is due to the fact that the greedy heuristic unequivocally selects a function to execute at each stage (ties are resolved arbitrarily).

3. Algorithm Evaluation

3.1. *Construction of Transition Probabilities*

Recall that $p_{ij}(f_m(i,\underline{y})) = P(x_{k+1} = j \mid x_k = i, \underline{y}, f_m(i,\underline{y}))$ denotes the transition probability that the next goal state is j given that the current goal state is i and that a control function $f_m(i,\underline{y})$ is applied. Therefore, the transition probability depends on the current goal state, the successor goal state, and the function applied (which also depends on the current state). Generally, $p_{ij}(f_m(i,\underline{y}))$ values can be inferred from historical data. For illustrative purposes, however, we generate the transition probabilities as follows.

The prior probability $P(x_{k+1} = j \mid x_k = i)$ is viewed as a random variable, which is governed by various unforeseen events in the system such that $\sum_j P(x_{k+1} = j \mid x_k = i) = 1$. Furthermore, we can reasonably assume that, in the absence of any of unforeseen events, the amount of resources and time committed to transform the system state to the next state, as well as the distance between the two states determine the probability of transition to the intended state, $g_j(r(f_m), d(f_m), \| j-i \|^{-1})$. Based on Bayes' rule and the total probability theorem [14], we assume the following relation for transition probabilities:

$$p_{ij}(j \mid i, \underline{y}, f_m(i, \underline{y})) = \frac{g_j(r(f_m), d(f_m), \| j-i \|^{-1}) P(j \mid i)}{\sum_j g_j(r(f_m), d(f_m), \| j-i \|^{-1}) P(j \mid i)} \tag{14}$$

where $\|\bullet\|$ denotes a valid norm of the arguments. The transition probabilities for the given example are generated using (14) and are listed in Table III of Appendix.

3.2. *Computational Experiments*

Consider the scenario introduced in section 1, where the desired goal state $x_k = 11$ is to be achieved on or before a desired deadline of $d = 6$ time units using at most $r = 5$ units of resource. It is assumed that the initial goal state $x_k = 1$, so that $\underline{s}_0 = (1,5,6)$, as shown in Figure 3. The available control functions $f_m(i, \underline{y}_k)$ and their reachable successor states for each goal state are as listed in Table IV of Appendix.

The DP-based optimal decision tree, highlighted in Figure 3, can be interpreted as follows. In the initial state (1,5,6), the possible actions are $\{f_1, f_2, f_3, f_5, f_6, f_7\}$. At this stage, the best control function is either f_1 or f_2. If the system chooses f_1, and if the next state (which could be either (1,2,2) or (2,2,2)) is (1,2,2), then the organization should stop, because any option it chooses (f_2 or f_3) will not lead to the desired state. However, if the next state is (2,2,2), the best control action is f_2. On the other hand, if the system selects f_2 at the initial state, and if the next state is (3,4,5), the best control action is f_5, and so on.

Next, the AO* algorithm with HEF $h_1(\underline{s})$ was applied for this example. The resulting decision tree is shown in Figure 4. Note that at $k = 0$, there are six feasible actions $\{f_1, f_2, f_3, f_5, f_6, f_7\}$. Among these, f_2 has the largest value of highest revised expected value $\sum_j p_{ij} h(\underline{s}_j)$ at the initial node. In particular, $\sum_j p_{ij} h(\underline{s}_j)$ for f_2 is $0.243(0.327) + 0.141(0.670) = 0.174$. Consequently, at this stage, the algorithm chooses f_2 as the control function. The partial tree through function f_2 is then expanded. The search proceeds further as follows. First, partial tree through function f_2 is traced to its terminal nodes. The goal state (1,4,5) has the smallest HEF, and therefore it is expanded first. It is determined that at this point f_2 yields the highest revised expected value at (1,5,6). At the other possible goal state (3,4,5), function f_5 is deemed as the best choice. At each of the new nodes, the previous step is repeated. The final revised value is 0.022, the optimal value as previously obtained via DP solution.

The graphs at each cycle of AO* are shown in Figure 4, and the optimal decision tree is as revealed at the last cycle. In this example, in the initial state (1,5,6), the best control function is f_2. If the next state is (1,4,5), the best control action is f_2, and so on.

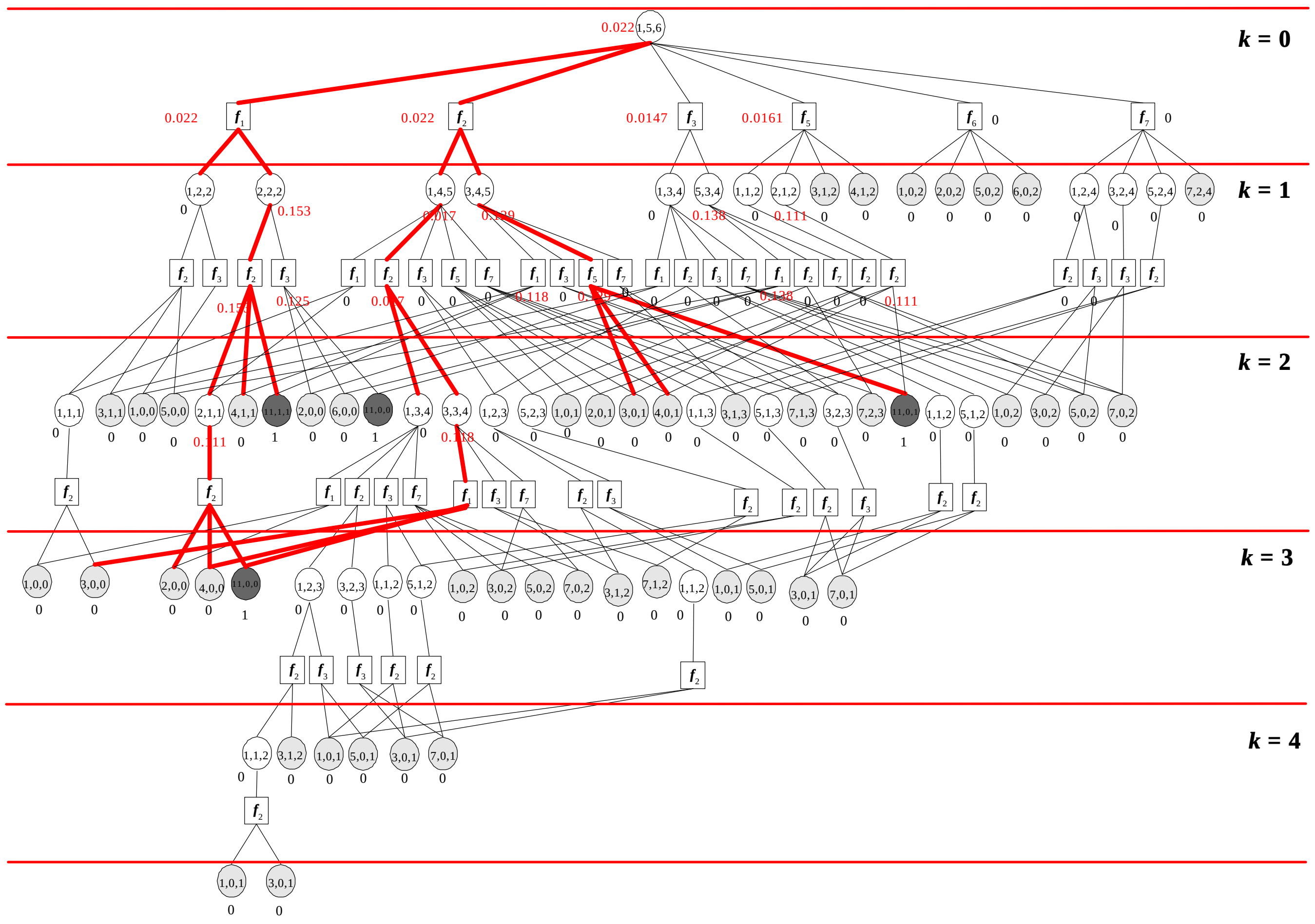


Figure 3. State Transitions with Selected Sets of Policies via DP-Algorithm

Embedded in the optimal decision tree are a number of different sequences in which a team's goals can be realized. For instance, the DP identifies all four of the best option sequences (f_2, f_5), (f_2, f_2, f_1), (f_1, f_2), or (f_1, f_2, f_2), that the system can take to arrive at the same expected success probability. The AO* recognizes the first two. Moreover, at any system state, the organization can opt to choose a control function other than the best, and be able to predict the consequences of the selected strategy. If necessary, all of the algorithms can easily carry the second (or even third, fourth, etc.) at each state in all stages with little additional computational cost and at a slightly increased storage cost.

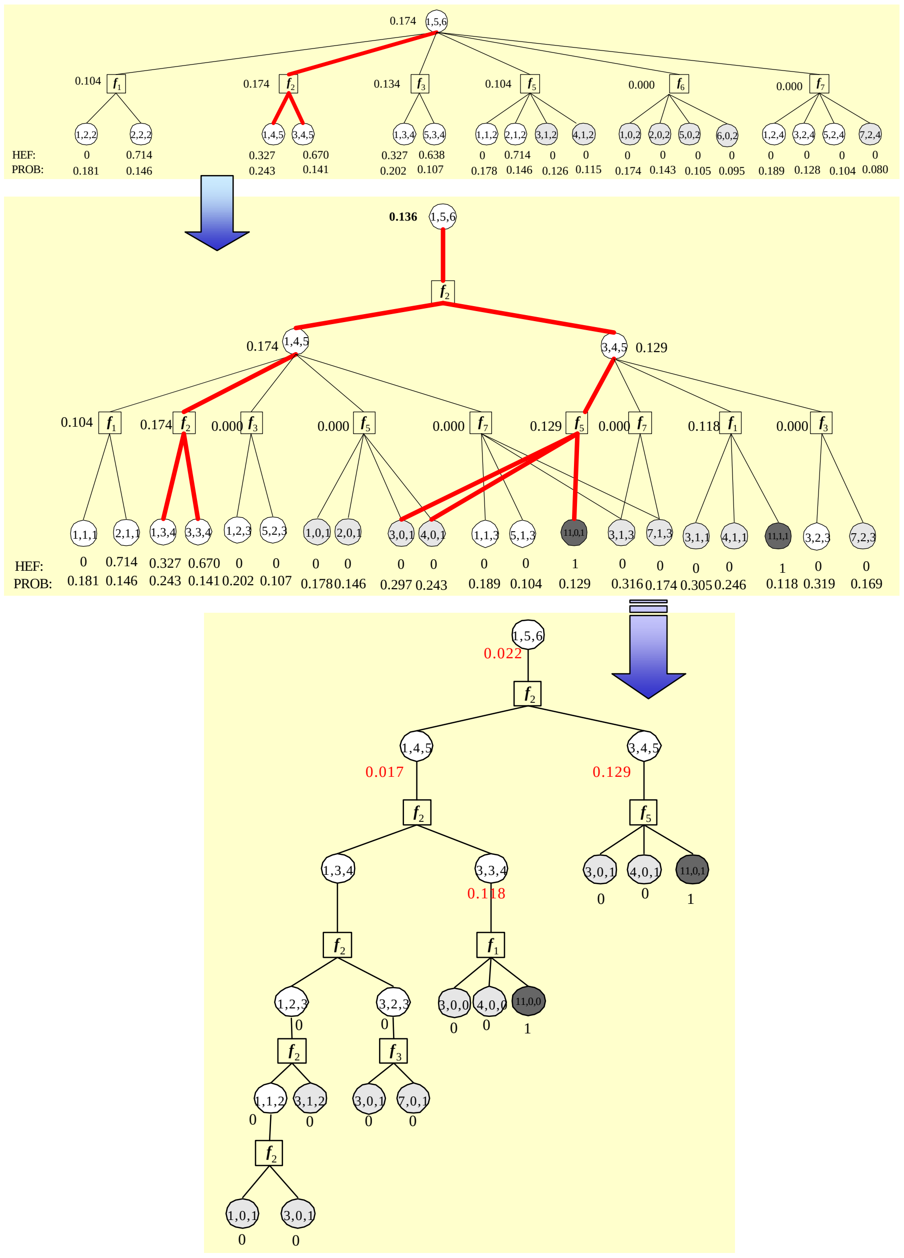


Figure 4. State Transitions with Selected Sets of Policies via AO* Search Procedure with $h_1(S)$

3.3. *Advantages, Shortcomings, and the Alternatives for the AO* Algorithm*

Even for the small-size problems, one can appreciate the advantage of AO* over DP in alleviating the computational explosion by avoiding the exploration of the entire set of solution trees. In this example, the AO* search generates only 19 nodes, as opposed to the DP that would have required 78 nodes (with $M = 7$, $|X| = 11$, and $Q = 1$, DP requires $O(77)$ nodes). The number of backtracks for AO* with HEF 1 is 0 for this example. In general, AO* with an admissible and tight (close bound of the value-to-go) HEF significantly reduces the computational burden of the DP, while still providing the optimal solution tree.

As Q increases (that is as (r,d) pairs increase), however, the computational burden of backtracking in AO* is considerable. This is one of the inherent shortcomings of the approach.

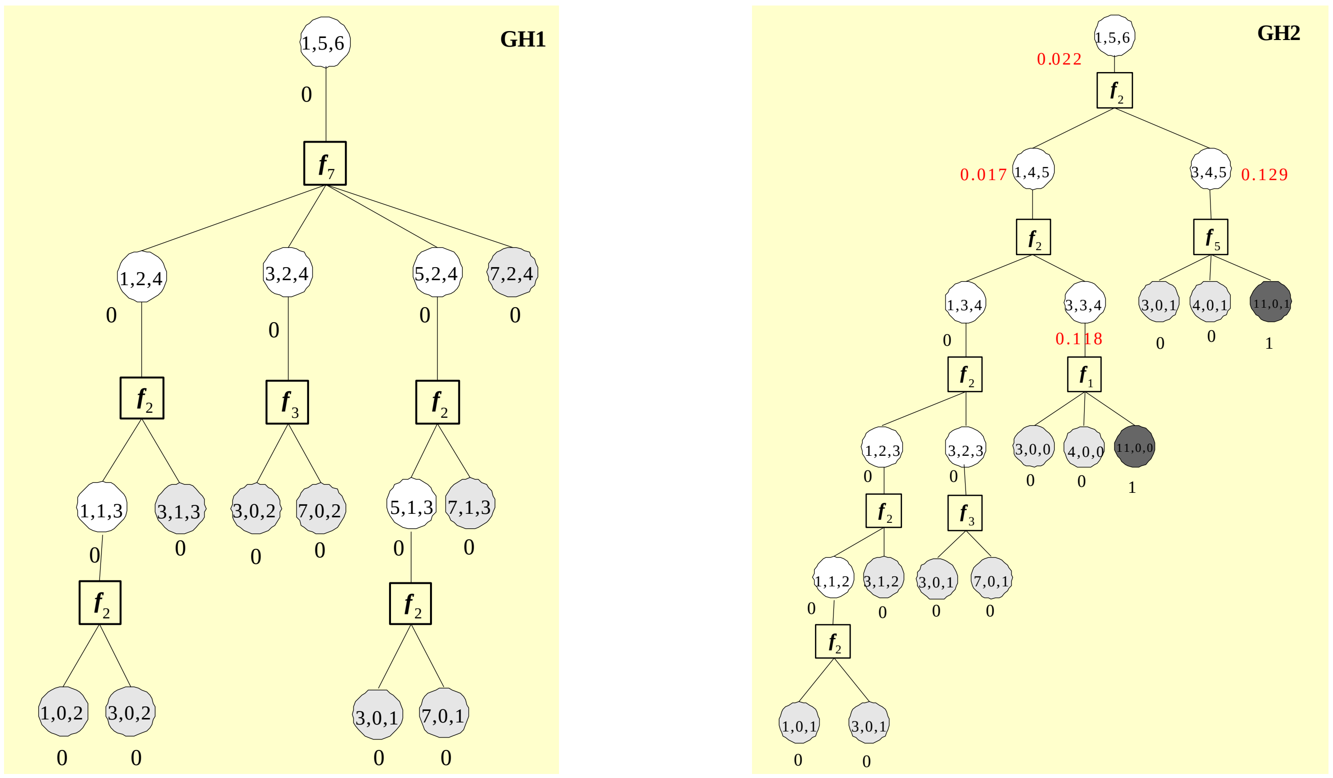
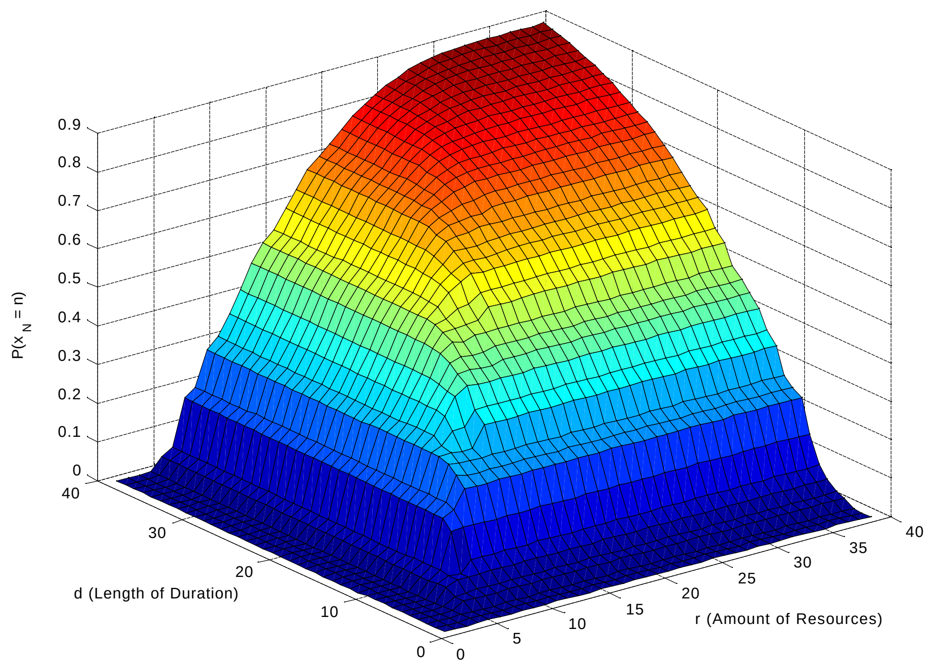


Figure 5. Functional Strategies Obtained via GH1 and GH2

The first greedy heuristic (GH1) tends to seize the immediate rewards at the expense of long-term gain, and subsequently suffers from the consequences. See Figure 5. Fortunately, this is not always the case in general. As will be shown in the next subsection, this approach results in acceptable control strategies. The second greedy heuristic (GH2) alleviates the propensity to seize the immediate rewards by using HEFs to account for future values. The greedy search chooses a node with the largest HEF at each step. It differs from AO*, in the sense that it performs limited search with no backtracking. In our example, GH2 results in the same solution as that of DP and AO*; see Figure 5. The second greedy heuristic GH2 typically results in near-optimal control strategies.

3.4. *Optimal Choices of Resources and Duration Lengths*

Figure 6 illustrates the effects of r and d on the probability of success $P(x_N = 11)$. The shape of the surface plot indicates that in order to achieve higher probability of success, the system needs to commit higher r and d. The flat lines in the contour plot (below), however, indicate that for each value of r (or d), there is only a limited set of options for the system in term of the other variable. This result can be illustrated further via the following figures.



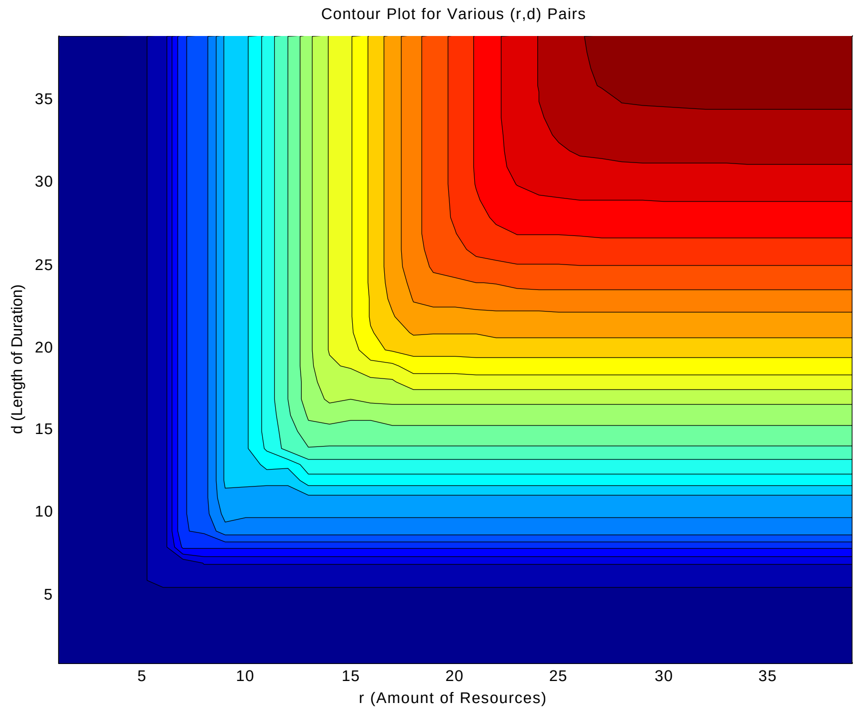


Figure 6. Surface and Contour Plots of $P(x_N = n)$ for Various (r, d) Pairs via GH2

From plots in Figure 7 and Figure 8, one can readily observe the following trends. The success probability of reaching the desired goal states, $P(x_N = 11)$, increases as r and d increase. As d steadily increases from 1 to 40 time units, the increase in $P(x_N = 11)$ is moderated by the availability of resource units, r. In particular, for GH2, if the system has r = 25 resource units, the system need not commit beyond d = 29 time units because the probability of success saturates at 0.72. On the other hand, if the system has r = 50 resource units, the system can achieve even higher $P(x_N = 11)$ of 0.93, by increasing d to about 40 time units. See Figure 8.

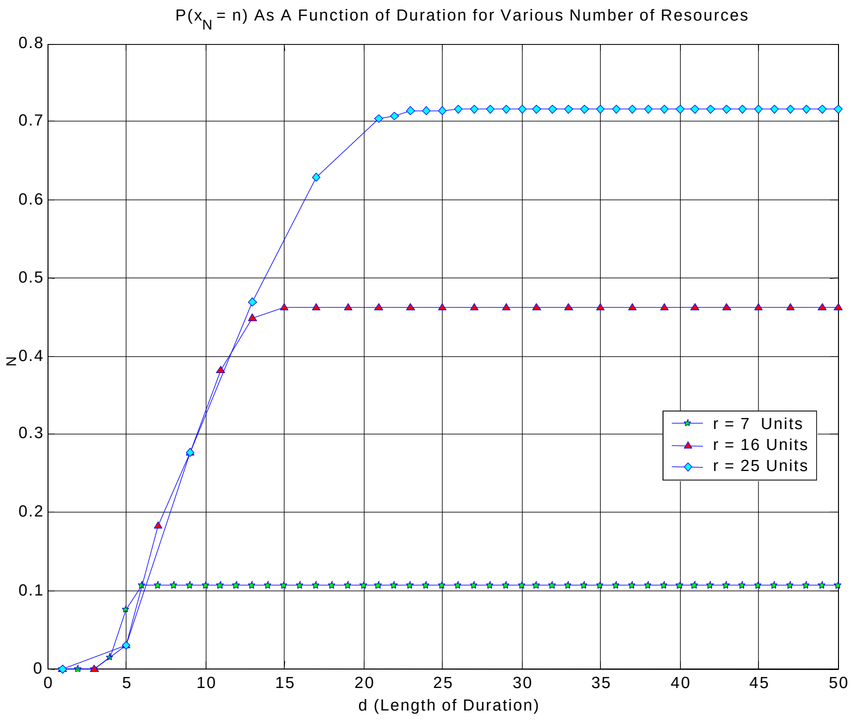
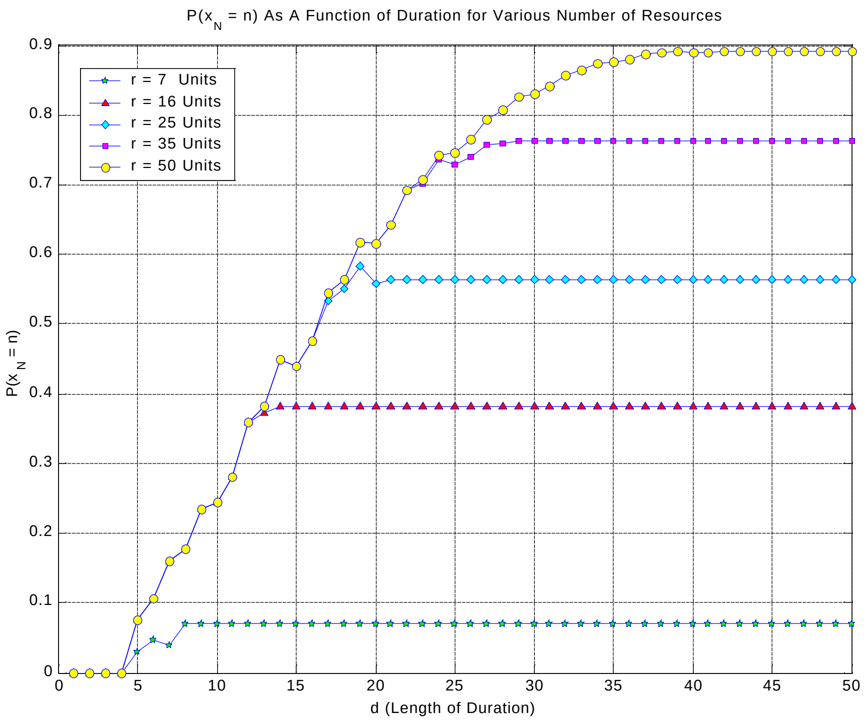


Figure 7. $P(x_N = n)$ **as a Function of Duration Lengths for Various Resource Units for AO***

One can also observe the degree of suboptimality of the greedy heuristic strategies compared to those of the AO*. The $P(x_N = 11)$ of the AO* functional strategies reaches 0.46 when d = 15 time units for r = 16 units, whereas the GH2 never attains 0.46 for the same r value. It saturates at 0.45. The same situation happens for GH1, which saturates at 0.38. For r = 25 units, however, the performance of AO* and GH2 algorithms resemble one another very closely. However, GH1 lags considerably. It appears that the degree of suboptimality of GH2 to AO* decreases as the actual value of $P(x_N = 11)$ increases. This may be attributed to the HEF being able to closely follow the actual value-to-go. For all practical purposes, GH2 is superior to GH1.



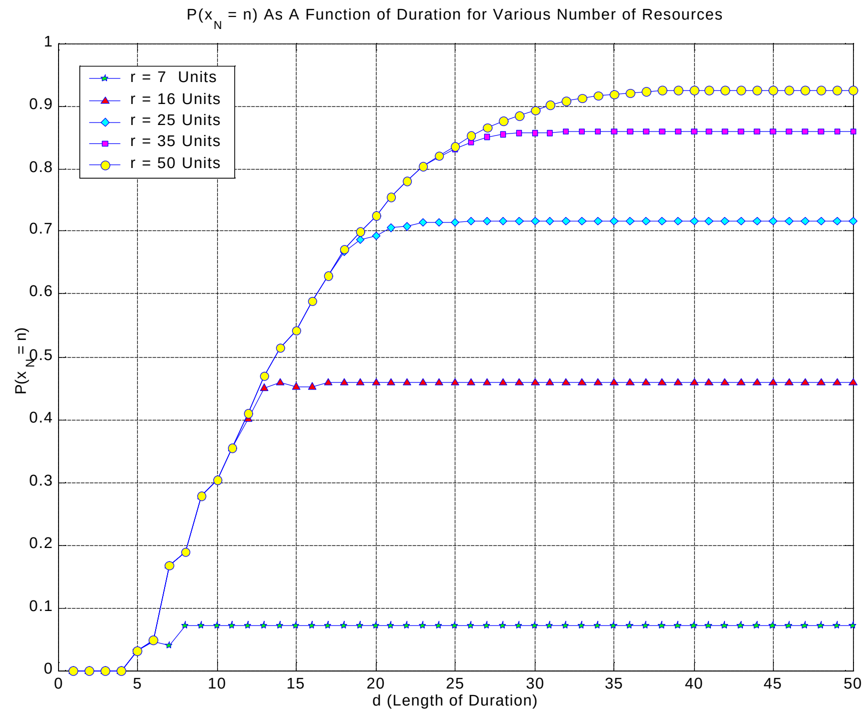


Figure 8. $P(x_N = n)$ **as a Function of** d **for** r **for GH1 and GH2, respectively**

Analogous observations can be made on the impact of d. See Figure 9. As the number of committed resources r increases steadily from 1 to 50 units, smaller duration length d saturates $P(x_N = 11)$ at a lower value. For example, if the system can only afford d = 7 time units, the system need not expend beyond r = 10 units of resource. The small value of d limits the probability of success at about 0.17, at best. One can also observe that d = 35 time units are about the optimal value for the system, with a maximum probability value of 0.93. Even if the system has more time, say d = 50 time units, the probability is about the same as that for d = 35 time units.

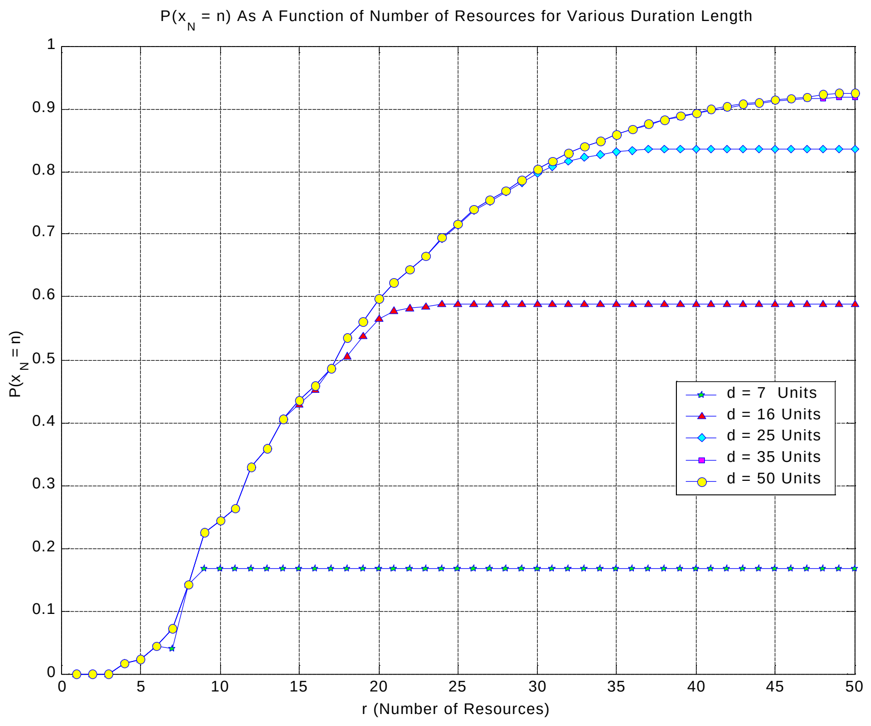


Figure 9. $P(x_N = n)$ **as a Function of** r **for Various Value of** d **for Greedy Heuristic 2 (GH2)**

It is worth noting that there is an inherent bound on the probability of success for a problem. For example, even if the system has unlimited r and d (e.g. beyond (100,100)), the value of $P(x_N = 11)$ $\cong$ 0.93. That is, in our example, the problem itself has an inherent failure probability of around 0.07 due to execution failures and random state shifts during the execution of control functions.

4. Conclusions and Future Work

This paper adopted a Markov decision process (MDP) framework as an underlying model for the problem, and introduced an explicit connection between the traditional planning routines (in particular AND/OR graph representations, which exclude uncertainties) and the MDP-based approach. The objective of the MDP is to devise courses of actions (plans or policies) with a high probability of success. In the future, we will augment the approach to include forbidden system states in our problem formulation. That is, we seek to find control strategies, which guarantee a high probability of success in reaching the desired goal states, while avoiding the forbidden goal states.

The paper introduced problem-specific HEFs into the search algorithms to address the computational adequacy of MDPs as a planning model. In particular, the approach exploited the goal structure to significantly reduce the state space. The integration of the new HEFs and heuristic search has enabled us to find verifiable optimal solutions to problems, which are intractable with the DP approach. It appears that the greedy heuristic, GH2, is the preferred algorithm for large size problems.

It is evident that the challenge is to find even better HEFs for the problem. In [16], it is suggested that the most useful HEFs should be able to solve at least a special case of the problem at hand, computationally efficient, and take into account the problem domain specifics. For example, an admissible HEF may be derived by assuming unlimited r and d. In addition, the AO*-based greedy search (GH2), which yields promising results, can be further improved using rollout strategies [3], [19].

Bibliography

[1] Bertsekas, D. P., and J. N. Tsitsiklis. (1991). "An Analysis of Stochastic Shortest Path Problems", *Mathematics of Operations Research*, Vol. 16, 580 – 595.

[2] Bertsekas, D. P. (1995). *Dynamic programming and optimal control, Vol. I and II*. Belmont, MA: Athena Scientific.

[3] Bertsekas, D. P., J. N. Tsitsiklis, and C. Wu. (1997). "Rollout Algorithms for Combinatorial Optimization," *Heuristics*, Vol. 3, 245 – 262.

[4] Bertsekas, D. P. (1999). *Nonlinear Programming*. Belmont, MA: Athena Scientific.

[5] Boutilier, C., Brafman, R. I., & Geib, C. (1997). "Prioritized goal decomposition of Markov decision processes: Toward a synthesis of classical and decision theoretic planning," *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pp. 1156 – 1162, Nagoya, Japan.

[6] Boutilier, C., Brafman, R. I., & Geib, C. (1998). "Structured Reachability Analysis for Markov Decision Processes," *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pp. 24 – 32 Madison, WI.

[7] Cao, Tiehua and A.C. Sanderson. (1998). "AND/OR Net Representation for Robotic Task Sequence Planning", *IEEE Transaction On Systems, Man, and Cybernetics – Part C: Applications and Reviews.* Vol. 28(2).

[8] Das, T. P., A. Gosavi, et al. (1999). "Solving Semi-Markov Decision Problems Using Average Reward Reinforcement Learning", *Management Science*. Vol. 45(4), 560 – 574.

[9] Feinberg, E. A., and A. Shwartz. (2002). *Handbook of Markov Decision Processes, Methods and Applications*. Norwell, MA: Kluwer Academic Publishers.

[10] Levchuk, Y. N. (2002). "The Art and Science of Organizational Design: Theory, Algorithms, and Applications," *working paper*.

[11] Littman, Michael L. (1997). "Probabilistic Propositional Planning: Representations and Complexity", *Proceedings of the 14th National Conference on Artificial Intelligence* (*AAAI-97*).

[12] Littman, Michael L., and Stephen M. Majercik. (1997). "Large-Scale Planning Under Uncertainty: A Survey", *Workshop on Planning and Scheduling for Space*, pages 27:1 – 8.

[13] Murphy, Kevin P. (2000). *A Survey of POMDP Solution Techniques*. Technical Report, University of California at Berkeley.

[14] Papoulis, Athanasios. (1991). *Probability, random variables, and stochastic processes*. New York, NY: McGraw-Hill.

[15] Parr, Ronald E. (1998). *Hierarchical Control and Learning for Markov Decision Processes*. Doctoral Dissertation, University of California at Berkeley.

[16] Pattipati, K.R., and M. G. Alexandridis. (1990). "Application of Heuristic Search and Information Theory to Sequential Fault Diagnosis," *IEEE Transactions On Systems, Man, and Cybernetics*. Vol. 20(4), 872 - 887.

[17] Pearl, J. (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, MA: Addison – Wesley Publishing Company, Inc.

[18] Pearl, J. (2000). *Causality: Models, Reasoning, and Inference*. Cambridge, NY: Cambridge University Press.

[19] Tu, Fang, and K. R. Pattipati. (2002). “Rollout Strategies for Sequential Fault Diagnosis,” *Working paper*.
[20] Yazia-Pekergin, N., J. M. Vincent. (1991). “Stochastic Bounds on Execution Times of Parallel Programs,” *IEEE Transactions on Software Engineering*. Vol. 17(10), 1005 –1012.

Appendix

i	m	$p_{i1}(f_m)$	$p_{i2}(f_m)$	$p_{i3}(f_m)$	$p_{i4}(f_m)$	$p_{i5}(f_m)$	$p_{i6}(f_m)$	$p_{i7}(f_m)$	$p_{i8}(f_m)$	$p_{i9}(f_m)$	$p_{i,10}(f_m)$	$p_{i,11}(f_m)$
1	1	0.1809	0.1459	0.1254	0.1109	0.1049	0.0904	0.0904	0.0812	0	0	0.07
	2	0.2434	0.1819	0.1407	0.1115	0.1115	0.0704	0.0704	0.0704	0	0	0
	3	0.2021	0.1592	0.1292	0.1189	0.1068	0.092	0.0729	0.0729	0	0	0.046
	4	0.189	0.1488	0.1275	0.1129	0.1039	0.0934	0.0804	0.0804	0	0	0.0637
	5	0.1776	0.1456	0.1262	0.1147	0.1051	0.0931	0.0857	0.0857	0	0	0.0663
	6	0.1744	0.1428	0.1263	0.1132	0.1049	0.0947	0.0886	0.0816	0	0	0.0733
	7	0.189	0.1488	0.1275	0.1129	0.1039	0.0934	0.0804	0.0804	0	0	0.0637
	8	0.171	0.1415	0.1251	0.1121	0.1055	0.0978	0.0883	0.0826	0	0	0.0761
	9	0.1776	0.1456	0.1262	0.1147	0.1051	0.0931	0.0857	0.0857	0	0	0.0663
2	1	0	0.3164	0	0.2194	0	0.1836	0	0.1582	0	0	0.1224
	2	0	0.3825	0	0.2211	0	0.1752	0	0.1106	0	0	0.1106
	3	0	0.3461	0	0.2212	0	0.183	0	0.1249	0	0	0.1249
	4	0	0.3348	0	0.2258	0	0.1841	0	0.1424	0	0	0.1129
	5	0	0.3107	0	0.2208	0	0.1839	0	0.1499	0	0	0.1347
	6	0	0.3073	0	0.2225	0	0.1848	0	0.1562	0	0	0.1292
	7	0	0.3348	0	0.2258	0	0.1841	0	0.1424	0	0	0.1129
	8	0	0.3021	0	0.221	0	0.1865	0	0.156	0	0	0.1345
	9	0	0.3107	0	0.2208	0	0.1839	0	0.1499	0	0	0.1347
10	1	0	0	0	0	0	0	0	0	0	0.5535	0.4465
	2	0	0	0	0	0	0	0	0	0	0.5723	0.4277
	3	0	0	0	0	0	0	0	0	0	0.5594	0.4406
	4	0	0	0	0	0	0	0	0	0	0.5595	0.4405
	5	0	0	0	0	0	0	0	0	0	0.5495	0.4505
	6	0	0	0	0	0	0	0	0	0	0.5498	0.4502
	7	0	0	0	0	0	0	0	0	0	0.5595	0.4405
	8	0	0	0	0	0	0	0	0	0	0.5471	0.4529
	9	0	0	0	0	0	0	0	0	0	0.5495	0.4505

Table III. Transition probabilities $p_{ij}(f_m)$ of $i = 1,2,\text{ and }10$ for each $f_m \in F(i)$

x_k	f_m	x_{k+1}	x_k	f_m	x_{k+1}
1	f_1	1,2	5	f_1	5,6,11
	f_2	1,3		f_2	5,7
	f_3	1,5		f_5	5,6,7,8,11
	f_5	1,2,3,4		f_6	5,6,11
	f_6	1,2,5,6		f_7	5,7
	f_7	1,3,5,7		f_8	5,6,7,8,11
	f_8	1,2,3,4,5,6,7,8,11	6	f_2	6,8,11
2	f_2	2,4,11		f_5	6,8,11
	f_3	2,6,11		f_7	6,8,11
	f_5	2,4,11		f_8	6,8,11
	f_6	2,6,11	7	f_1	7,8,11
	f_7	2,4,6,11		f_4	7,9,11
	f_8	2,4,6,8,11		f_5	7,8,11
3	f_1	3,4,11		f_6	7,8,11
	f_3	3,7		f_8	7,8,11
	f_5	3,4,11		f_9	7,8,11
	f_6	3,4,7,8,11	8	f_4	8,10,11
	f_7	3,7		f_9	8,10,11
	f_8	3,4,7,8,11	9	f_1	9,10,11
4	f_3	4,8,11		f_5	9,10,11
	f_6	4,8,11		f_6	9,10,11
	f_7	4,8,11		f_8	9,10,11
	f_8	4,8,11	10	f_9	10,11

Table IV. Available Control Functions and State Transitions at each Goal State