April 2005 **Report LIDS 2646**

Rollout Algorithms for Constrained Dynamic Programming[1]

by

Dimitri P. Bertsekas[2]

Abstract

The rollout algorithm is a suboptimal control method for deterministic and stochastic problems that can be solved by dynamic programming. In this short note, we derive an extension of the rollout algorithm that applies to constrained deterministic dynamic programming problems, and relies on a suboptimal policy, called base heuristic. Under suitable assumptions, we show that if the base heuristic produces a feasible solution, the rollout algorithm also produces a feasible solution, whose cost is no worse than the cost corresponding to the base heuristic.

[1] Supported by NSF Grant ECS-0218328.

[2] Dept. of Electrical Engineering and Computer Science, M.I.T., Cambridge, Mass., 02139.

1. INTRODUCTION

We consider a deterministic optimal control problem involving the system

$$x_{k+1} = f_k(x_k, u_k), \qquad k = 0, \ldots, N-1,$$

where x_k and u_k are the state and control at time k, respectively, taking values in some sets, which may depend on k, and f_k is some function. The initial state is given and is denoted by x_0. Each control u_k must be chosen from a finite constraint set $U_k(x_k)$ that depends on the current state x_k. A sequence of the form

$$T = (x_0, u_0, x_1, u_1, \ldots, u_{N-1}, x_N),$$

where

$$x_{k+1} = f_k(x_k, u_k), \qquad u_k \in U_k(x_k), \qquad k = 0, 1, \ldots, N-1,$$

is referred to as a *trajectory*. In our terminology, a trajectory is complete in the sense that it starts at the given initial state x_0, and ends at some state x_N after N stages. We will also refer to *partial trajectories*, which are subsets of complete trajectories, involving fewer than N stages and consisting of stage-contiguous states and controls.

The cost of the trajectory $T = (x_0, u_0, x_1, u_1, \ldots, u_{N-1}, x_N)$ is

$$V(T) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k), \tag{1.1}$$

where g_k, $k = 0, 1, \ldots, N$, are given functions. The problem is to find a trajectory T that minimizes $V(T)$ subject to the constraint

$$T \in C, \tag{1.2}$$

where C is a given set of trajectories.

An optimal solution of this problem can be found by an extension of the dynamic programming algorithm (DP for short), but the associated computation can be overwhelming. It is much greater that the computation for the corresponding unconstrained problem where the constraint $T \in C$ is absent. This is true even in the special case where C is specified in terms of a finite number of constraint functions that are time-additive, i.e., $T \in C$ if

$$g_N^m(x_N) + \sum_{k=0}^{N-1} g_k^m(x_k, u_k) \leq b^m, \qquad m = 1, \ldots, M, \tag{1.3}$$

where g_k^m, $k = 0, 1, \ldots, N$, and b^m, $m = 1, \ldots, M$, are given functions and scalars, respectively. The literature contains several proposals for suboptimal solution of the problem for the case where the constraints are of

the form (1.3). Most of these proposals are cast in the context of the constrained and multiobjective shortest path problems; see e.g., Jaffe [Jaf84], Martins [Mar84], Guerriero and Musmanno [GuM01], and Stewart and White [StW91], who also survey earlier work.

In this paper, we propose a suboptimal approach, which may be viewed as an extension of the rollout algorithm, a method that has been used with considerable success for general unconstrained DP problems (including stochastic and minimax); see Tesauro and Galperin [TeG96], Bertsekas and Tsitsiklis [Ber96], Bertsekas, Tsitsiklis, and Wu [BTW97], Bertsekas [Ber97], Christodouleas [Chr97], Bertsekas and Castanon [BeC99], Secomandi [Sec00], [Sec01], [Sec03], Bertsimas and Demir [BeD02], Ferris and Voelker [FeV02], [FeV04], McGovern, Moss, and Barto [MMB02], Savagaonkar, Givan, and Chong [SGC02], Bertsimas and Popescu [BeP03], Guerriero and Mancini [GuM03], Tu and Pattipati [TuP03], Wu, Chong, and Givan [WCG03], Chang, Givan, and Chong [CGC04], Meloni, Pacciarelli, and Pranzo [MPP04], Tu, Pham, Luo, Pattipati, and Willett [TPL04], and Yan, Diaconis, Rusmevichientong, and Van Roy [YDR05]. These works discuss a broad variety of applications and case studies, and generally report positive computational experience.

We note that it is possible to reformulate the constrained problem of this paper into a format that is suitable for application of the rollout algorithm for discrete deterministic problems, given in [BTW97] and [Ber01]. In particular, we may redefine the state at stage k to be the partial trajectory

$$y_k = (x_0, u_0, x_1, \ldots, u_{k-1}, x_k),$$

which evolves according to

$$y_{k+1} = \big(y_k, u_k, f_k(x_k, u_k)\big).$$

The problem then becomes to find a control sequence that minimizes a suitable terminal cost function $F(y_N)$ subject to the constraint $y_N \in C$. This is a format to which the rollout approach of [BTW97] applies, and yields algorithms and results that are quite similar to the ones obtained in this paper. However, this line of development, while conceptually valuable, is complicated and unintuitive. It also does not lend itself to discussion and development of variants of the main algorithm, so in this paper, we have chosen a more direct extension of the rollout approach that is specially adapted to constrained problems.

The rollout algorithm has some additional special properties when applied to discrete unconstrained deterministic problems, as discussed in Bertsekas, Tsitsiklis, and Wu [BTW97]; see also the author's textbook [Ber01], which provides an extensive discussion. The rollout algorithm makes use of a suboptimal algorithm, called *base heuristic*. If the base heuristic is a DP policy, i.e., it generates state/control sequences via feedback functions μ_k, $k = 0, 1, \ldots, N-1$, so that at any state x_k, it applies the control $\mu_k(x_k)$, then the rollout algorithm can be viewed as a policy improvement step of the policy iteration method. Based on generic results for policy iteration, it then follows that the rollout algorithm's cost is no worse than the one

of the base heuristic. This cost improvement property has been extended in [BTW97] to base heuristics that are not DP policies, but rather satisfy an assumption called *sequential improvement*, which will be discussed shortly within the constrained context of this paper.

Our extension of the rollout algorithm to discrete deterministic problems with the constraints (1.2), also uses a base heuristic, which has the property that given any state x_k at stage k, it produces a partial trajectory

$$H(x_k) = (x_k, u_k, x_{k+1}, u_{k+1}, \ldots, u_{N-1}, x_N),$$

that starts at x_k and satisfies

$$x_{i+1} = f_i(x_i, u_i), \qquad u_i \in U_i(x_i), \qquad i = k, \ldots, N-1.$$

The cost corresponding to the partial trajectory $H(x_k)$ is denoted by $\tilde{J}(x_k)$:

$$\tilde{J}(x_k) = g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, u_i).$$

Thus, given a partial trajectory that starts at the initial state x_0 and ends at a state x_k, the base heuristic can be used to complete this trajectory by concatenating it with the partial trajectory $H(x_k)$. The trajectory thus obtained is not guaranteed to be feasible (i.e., it may not belong to C), but we assume throughout that *the state/control trajectory generated by the base heuristic starting from the given initial state x_0 is feasible*, i.e.,

$$H(x_0) \in C.$$

Finding a base heuristic with this property may not be easy, but we do not address this question in this paper. It appears, however, that for many problems of interest, there are natural base heuristics that satisfy this feasibility requirement; this is true in particular when C is specified by Eq. (1.3) with only one constraint function ($M = 1$), in which case a feasible base heuristic can be obtained (if at all possible) by ordinary (unconstrained) DP, using as cost function either the constraint function or an appropriate weighted sum of the cost and constraint functions.

The rollout algorithm that we propose for constrained DP problems is not much more complicated or computationally demanding than the one for unconstrained DP problems (as long as checking feasibility of a trajectory T, i.e., $T \in C$, is not computationally demanding). We will show under an appropriate extension of the notion of sequential improvement, that the rollout algorithm produces a feasible solution, whose cost is no worse than the cost corresponding to the base heuristic. We note, however, that it does not appear possible to relate our rollout algorithm to concepts of policy iteration, and in fact we are not aware of an analog of the policy iteration method for constrained DP problems, even when the constraint set C is specified by inequality constraints with constraint functions that are additive over time, as in Eq. (1.3).

Also, our rollout algorithm makes essential use of the deterministic character of the problem, and does not admit a straightforward extension to stochastic or minimax problems.

In the next section, we introduce our algorithm, and in Section 3 we establish its cost improvement properties. In Section 4, we discuss some variants and extensions of the algorithm.

2. THE ROLLOUT ALGORITHM

The rollout algorithm proposed in this paper starts at stage 0 and sequentially proceeds to the last stage. At stage k, it maintains a partial trajectory

$$T_k = (x_0, \overline{u}_0, \overline{x}_1, \ldots, \overline{u}_{k-1}, \overline{x}_k) \tag{2.1}$$

that starts at the given initial state x_0, and is such that

$$\overline{x}_{i+1} = f_i(\overline{x}_i, \overline{u}_i), \qquad \overline{u}_i \in U_i(\overline{x}_i), \qquad i = 0, 1, \ldots, k-1,$$

where $\overline{x}_0 = x_0$. The algorithm starts with the partial trajectory T_0 that consists of just the initial state x_0.

For each $k = 0, 1, \ldots, N-1$, and given the current partial trajectory T_k, it forms for each control $u_k \in U_k(\overline{x}_k)$, a (complete) trajectory $T_k^c(u_k)$ by concatenating:

(1) The current partial trajectory T_k.

(2) The control u_k and the next state $x_{k+1} = f_k(\overline{x}_k, u_k)$.

(3) The partial trajectory $H(x_{k+1})$ generated by the base heuristic starting from x_{k+1}.

Then, it forms the subset $\overline{U}_k(\overline{x}_k)$ of controls u_k for which $T_k^c(u_k)$ is feasible, i.e., $T_k^c(u_k) \in C$. The algorithm then selects from $\overline{U}_k(\overline{x}_k)$ a control $\overline{u}_k$ that minimizes over $u_k \in \overline{U}_k(\overline{x}_k)$†

$$g_k(\overline{x}_k, u_k) + \tilde{J}\big(f_k(\overline{x}_k, u_k)\big).$$

It then forms the partial trajectory T_{k+1} by adding $(\overline{u}_k, \overline{x}_{k+1})$ to T_k, where

$$\overline{x}_{k+1} = f_k(\overline{x}_k, \overline{u}_k).$$

† This is the only point in the paper where we use finiteness of $U_k(x_k)$ for all x_k, and by extension finiteness of $\overline{U}_k(\overline{x}_k)$. It guarantees that the minimum over $\overline{U}_k(\overline{x}_k)$ is well-defined. We may replace the finiteness assumption with the assumption that the minimum of $g_k(\overline{x}_k, u_k) + \tilde{J}\big(f_k(\overline{x}_k, u_k)\big)$ over $u_k \in \overline{U}_k(\overline{x}_k)$ is attained by some $\overline{u}_k$.

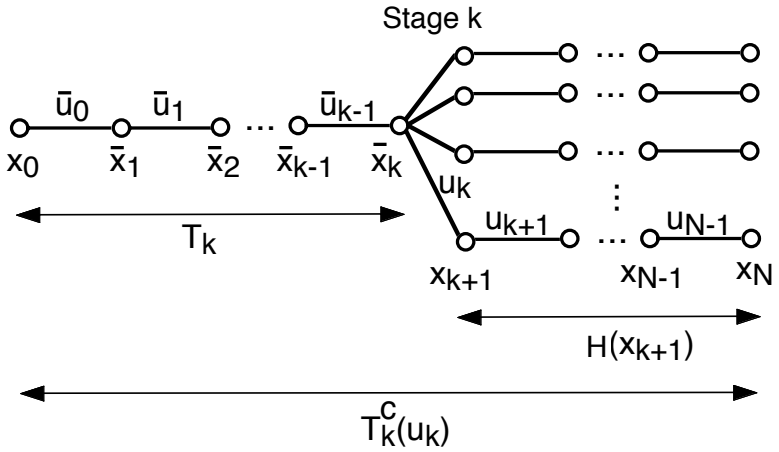

Figure 2.1. Illustration of the rollout algorithm. At stage k, and given the current partial trajectory T_k that starts at x_0 and ends at $\overline{x}_k$, it considers all possible next states $x_{k+1} = f_k(\overline{x}_k, u_k)$, $u_k \in U_k(\overline{x}_k)$, and runs the base heuristic starting at x_{k+1}. It then finds a control $\overline{u}_k \in U_k(\overline{x}_k)$ such that the corresponding trajectory

$$T_k^c(\overline{u}_k) = T_k \cup (\overline{u}_k) \cup H(\overline{x}_{k+1}),$$

where

$$\overline{x}_{k+1} = f_k(\overline{x}_k, \overline{u}_k),$$

is feasible and has minimum cost, and extends T_k by adding to it $(\overline{u}_k, \overline{x}_{k+1})$.

For a more compact definition of the rollout algorithm, let us use the union symbol $\cup$ to denote the concatenation of two or more partial trajectories. With this notation, we have

$$T_k^c(u_k) = T_k \cup (u_k) \cup H\big(f_k(\overline{x}_k, u_k)\big), \tag{2.2}$$

$$\overline{U}_k(\overline{x}_k) = \big\{u_k \mid u_k \in U_k(\overline{x}_k),\, T_k^c(u_k) \in C\big\}. \tag{2.3}$$

The rollout algorithm selects at stage k

$$\overline{u}_k \in \arg \min_{u_k \in \overline{U}_k(\overline{x}_k)} \Big[g_k(\overline{x}_k, u_k) + \tilde{J}\big(f_k(\overline{x}_k, u_k)\big)\Big], \tag{2.4}$$

and extends the current partial trajectory by setting

$$T_{k+1} = T_k \cup (\overline{u}_k, \overline{x}_{k+1}), \tag{2.5}$$

where

$$\overline{x}_{k+1} = f_k(\overline{x}_k, \overline{u}_k). \tag{2.6}$$

Figure 2.1 provides an illustration of the algorithm.

It is worth noting the similarity of this rollout algorithm with the unconstrained version given in [BTW97] and [Ber01]: the constraint set C enters only in Eq. (2.3), and if $\overline{U}_k(\overline{x}_k)$ is replaced by $U_k(\overline{x}_k)$ in Eq. (2.4), one obtains the rollout algorithm for unconstrained DP problems.

Note that it is possible that there is no control $u_k \in U_k(\overline{x}_k)$ that satisfies the constraint $T_k^c(u_k) \in C$ [i.e., the set $\overline{U}_k(\overline{x}_k)$ is empty], in which case the algorithm breaks down. We will show, however, that this cannot happen under some conditions that we now introduce.

Definition 2.1: The base heuristic is called *sequentially consistent* if whenever it generates a partial trajectory

$$(x_k, u_k, x_{k+1}, u_{k+1}, \ldots, u_{N-1}, x_N),$$

starting from state x_k, it also generates the partial trajectory

$$(x_{k+1}, u_{k+1}, x_{k+2}, u_{k+2}, \ldots, u_{N-1}, x_N),$$

starting from state x_{k+1}.

Thus, a base heuristic is sequentially consistent if, when started at intermediate states of a partial trajectory that it generates, it produces the same subsequent controls and states. If we denote by $\mu_k(x_k)$ the control applied by the base heuristic when at state x_k, then the sequence of control functions $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ can be viewed as a policy. The cost-to-go of this policy starting at state x_k, call it $J_k^\pi(x_k)$, and the cost $\tilde{J}(x_k)$ produced by the base heuristic starting from x_k need not be equal. However, they are equal when the heuristic is sequentially consistent, i.e.,

$$J_k^\pi(x_k) = \tilde{J}(x_k), \qquad \forall\ x_k,\ k,$$

because in this case, π generates the same partial trajectory as the base heuristic, when starting from the same state. It follows that, for a sequentially consistent base heuristic, we have

$$g_k\big(x_k, \mu_k(x_k)\big) + \tilde{J}\big(f_k\big(x_k, \mu_k(x_k)\big)\big) = \tilde{J}(x_k), \qquad \forall\ x_k,\ k. \tag{2.7}$$

Note that base heuristics that are of the "greedy" type tend to be sequentially consistent, as explained in [Ber01], Section 6.4. The next definition provides an extension of the notion of sequential consistency.

Definition 2.2: The base heuristic is called *sequentially improving* if for every x_k, the partial trajectory

$$H(x_k) = (x_k, u_k, x_{k+1}, u_{k+1}, \ldots, u_{N-1}, x_N),$$

has the following properties:

(1)
$$g_k(x_k, u_k) + \tilde{J}(x_{k+1}) \leq \tilde{J}(x_k). \tag{2.8}$$

(2) If for some partial trajectory of the form $(x_0, u_0, x_1, \ldots, u_{k-1}, x_k)$ we have

$$(x_0, u_0, x_1, \ldots, u_{k-1}, x_k) \cup H(x_k) \in C,$$

then

$$(x_0, u_0, x_1, \ldots, u_{k-1}, x_k, u_k, x_{k+1}) \cup H(x_{k+1}) \in C.$$

Note that if the base heuristic is sequentially consistent, it is also sequentially improving [cf. Eqs. (2.7) and (2.8)]. For another case where the base heuristic is sequentially improving, let the constraint set C consist of all trajectories $T = (x_0, u_0, x_1, u_1, \ldots, u_{N-1}, x_N)$ such that

$$g_N^m(x_N) + \sum_{k=0}^{N-1} g_k^m(x_k, u_k) \leq b^m, \qquad m = 1, \ldots, M,$$

[cf. Eq. (1.3)]. Let $\tilde{C}^m(x_k)$ be the value of the mth constraint function corresponding to the partial trajectory $H(x_k) = (x_k, u_k, x_{k+1}, u_{k+1}, \ldots, u_{N-1}, x_N)$, generated by the base heuristic starting from x_k, i.e.,

$$\tilde{C}^m(x_k) = g_N^m(x_N) + \sum_{i=k}^{N-1} g_i^m(x_i, u_i), \qquad m = 1, \ldots, M.$$

Then, it can be seen that the base heuristic is sequentially improving if in addition to Eq. (2.8), the partial trajectory $H(x_k)$ satisfies

$$g_k^m(x_k, u_k) + \tilde{C}^m(x_{k+1}) \leq \tilde{C}^m(x_k), \qquad m = 1, \ldots, M. \tag{2.9}$$

The reason is that if a trajectory of the form

$$(x_0, u_0, x_1, \ldots, u_{k-1}, x_k) \cup H(x_k)$$

belongs to C, then we have

$$\sum_{i=0}^{k-1} g_i^m(x_i, u_i) + \tilde{C}^m(x_k) \leq b^m, \qquad m = 1, \ldots, M,$$

and from Eq. (2.9), it follows that

$$\sum_{i=0}^{k-1} g_i^m(x_i, u_i) + g_k^m(x_k, u_k) + \tilde{C}^m(x_{k+1}) \leq b^m, \qquad m = 1, \ldots, M.$$

This implies that the trajectory

$$(x_0, u_0, x_1, \ldots, u_{k-1}, x_k, u_k, x_{k+1}) \cup H(x_{k+1})$$

belongs to C, thereby verifying property (2) of the definition of sequential improvement.

3. MAIN RESULTS

The essence of our main result is contained in the following proposition. To state compactly the result, consider the partial trajectory

$$T_k = (x_0, \overline{u}_0, \overline{x}_1, \ldots, \overline{u}_{k-1}, \overline{x}_k)$$

maintained by the rollout algorithm after k stages, the set of trajectories

$$\big\{T_k^c(u_k) \mid u_k \in \overline{U}_k(\overline{x}_k)\big\}$$

that are feasible [cf. Eqs. (2.2) and (2.3)], and the trajectory $\overline{T}_k^c$ within this set that corresponds to the control $\overline{u}_k$ chosen by the rollout algorithm, i.e.,

$$\overline{T}_k^c = T_k^c(\overline{u}_k) = T_k \cup (\overline{u}_k) \cup H(\overline{x}_{k+1}), \tag{3.1}$$

where

$$\overline{x}_{k+1} = f_k(\overline{x}_k, \overline{u}_k).$$

Proposition 3.1: Assume that the base heuristic is sequentially improving. Then for each k, the set $\overline{U}_k(\overline{x}_k)$ is nonempty, and

$$V\big(H(x_0)\big) \geq V(\overline{T}_0^c) \geq V(\overline{T}_1^c) \geq \cdots \geq V(\overline{T}_{N-1}^c) \geq V(\overline{T}_N^c),$$

where the trajectories $\overline{T}_k^c$ are given by Eq. (3.1) for all k.

Proof: Let the trajectory generated by the base heuristic starting from x_0 have the form

$$H(x_0) = (x_0, u'_0, x'_1, u'_1, \ldots, u'_{N-1}, x'_N),$$

and note that since $H(x_0) \in C$ by assumption, we have $u'_0 \in \overline{U}_0(x_0)$. Thus, the set $\overline{U}_0(x_0)$ is nonempty. Also, we have

$$V\big(H(x_0)\big) = \tilde{J}(x_0) \geq g_0(x_0, u'_0) + \tilde{J}\big(f_0(x_0, u'_0)\big),$$

where the inequality follows by the sequential improvement assumption [cf. Eq. (2.8)]. Since

$$\overline{u}_0 \in \arg \min_{u_0 \in \overline{U}_0(x_0)} \Big[g_0(x_0, u_0) + \tilde{J}\big(f_0(x_0, u_0)\big)\Big],$$

it follows by combining the preceding two relations and the definition of $\overline{T}_0^c$ [cf. Eq. (3.1)] that

$$V\big(H(x_0)\big) \geq g_0(x_0, \overline{u}_0) + \tilde{J}(\overline{x}_1) = V(\overline{T}_0^c),$$

while $\overline{T}_0^c$ is feasible by the definition of $\overline{U}_0(x_0)$.

The preceding argument can be repeated for the next stage, by replacing x_0 with $\overline{x}_1$, and $H(x_0)$ with $\overline{T}_0^c$. In particular, let $\overline{T}_0^c$ have the form

$$\overline{T}_0^c = (x_0, \overline{u}_0, \overline{x}_1, u'_1, x'_1, u'_2, \ldots, u'_{N-1}, x'_N).$$

Since $\overline{T}_0^c$ is feasible as noted earlier, we have $u'_1 \in \overline{U}_1(\overline{x}_1)$, so that $\overline{U}_1(\overline{x}_1)$ is nonempty. Also, we have

$$V\big(\overline{T}_0^c\big) = g_0(x_0, \overline{u}_0) + \tilde{J}(\overline{x}_1) \geq g_0(x_0, \overline{u}_0) + g_1(\overline{x}_1, u'_1) + \tilde{J}\big(f_1(\overline{x}_1, u'_1)\big),$$

where the inequality follows by the sequential improvement assumption. Since

$$\overline{u}_1 \in \arg \min_{u_1 \in \overline{U}_1(\overline{x}_1)} \Big[g_1(\overline{x}_1, u_1) + \tilde{J}\big(f_1(\overline{x}_1, u_1)\big)\Big],$$

it follows by combining the preceding two relations and the definition of $\overline{T}_1^c$ that

$$V(\overline{T}_0^c) \geq g_0(x_0, \overline{u}_0) + g_1(\overline{x}_1, \overline{u}_1) + \tilde{J}(\overline{x}_2) = V(\overline{T}_1^c),$$

while $\overline{T}_1^c$ is feasible by the definition of $\overline{U}_1(\overline{x}_1)$.

Similarly, the argument can be successively repeated for every k, to verify that $\overline{U}_k(\overline{x}_k)$ is nonempty and that $V(\overline{T}_{k-1}^c) \geq V(\overline{T}_k^c)$ for all k. **Q.E.D.**

Proposition 3.1 implies that the rollout algorithm generates at each stage k a feasible trajectory that is no worse than its predecessor in terms of cost. Since the starting trajectory is $H(x_0)$ and the final trajectory is

$$\overline{T}_N^c = (x_0, \overline{u}_0, \overline{x}_1, \ldots, \overline{u}_{N-1}, \overline{x}_N),$$

we have the following.

Proposition 3.2: If the base heuristic is sequentially improving, then the rollout algorithm produces a trajectory $(x_0, \overline{u}_0, \overline{x}_1, \ldots, \overline{u}_{N-1}, \overline{x}_N)$ that is feasible and has cost that is no larger than the cost of the trajectory generated by the base heuristic starting from x_0.

4. VARIANTS AND EXTENSIONS OF THE ROLLOUT ALGORITHM

We will now discuss some variations and extensions of the rollout algorithm of Section 2.

Rollout Without the Sequential Improvement Assumption

Let us consider the case where the sequential improvement assumption is not satisfied. Then, even if the base heuristic generates a feasible trajectory starting from the initial state x_0, it may happen that given the partial trajectory T_k, the set of controls $\overline{U}_k(\overline{x}_k)$ that corresponds to feasible trajectories $T_k^c(u_k)$ [cf. Eq. (2.3)] is empty, in which case the rollout algorithm cannot extend the trajectory T_k further. To bypass this difficulty, we propose a modification of the algorithm, called *fortified rollout algorithm*, which is an extension of an algorithm given in [BTW97] for the case of an unconstrained DP problem (see also [Ber01], Section 6.4).

The fortified rollout algorithm, in addition to the partial trajectory

$$T_k = (x_0, \overline{u}_0, \overline{x}_1, \ldots, \overline{u}_{k-1}, \overline{x}_k),$$

maintains a (complete) trajectory $\overline{T}$, which is feasible and agrees with T_k up to state $\overline{x}_k$, i.e., $\overline{T}$ has the form

$$\overline{T} = (x_0, \overline{u}_0, \overline{x}_1, \ldots, \overline{u}_{k-1}, \overline{x}_k, u_k, x_{k+1}, \ldots, u_{N-1}, x_N),$$

for some $u_k, x_{k+1}, \ldots, u_{N-1}, x_N$ such that $x_{i+1} = f_i(u_i, x_i)$ for $i = k, \ldots, N-1$, with $x_k = \overline{x}_k$. Initially, T_0 consists of the initial state x_0, and $\overline{T}$ is the trajectory $H(x_0)$, generated by the base heuristic starting from x_0. At stage k, the algorithm forms the subset $\tilde{U}_k(\overline{x}_k)$ of controls $u_k \in U_k(\overline{x}_k)$ such that $T_k^c(u_k) \in C$ *and*

$$\sum_{i=0}^{k-1} g_i(x_i, u_i) + g_k(x_k, u_k) + \tilde{J}\big(f_k(x_k, u_k)\big) \leq V(\overline{T}).$$

There are two cases to consider:

(1) The set $\tilde{U}_k(\overline{x}_k)$ is nonempty. Then, the algorithm selects from $\tilde{U}_k(\overline{x}_k)$ a control $\overline{u}_k$ that minimizes over $u_k \in \tilde{U}_k(\overline{x}_k)$

$$g_k(\overline{x}_k, u_k) + \tilde{J}\big(f_k(\overline{x}_k, u_k)\big).$$

It then forms the partial trajectory

$$T_{k+1} = T_k \cup (\overline{u}_k, \overline{x}_{k+1}),$$

where $\overline{x}_{k+1} = f_k(\overline{x}_k, \overline{u}_k)$, and replaces $\overline{T}$ with the trajectory

$$T_k \cup (\overline{u}_k) \cup H(\overline{x}_{k+1}).$$

(2) The set $\tilde{U}_k(\overline{x}_k)$ is empty. Then, the algorithm forms the partial trajectory T_{k+1} by concatenating T_k by the control/state pair (u_k, x_{k+1}) subsequent to $\overline{x}_k$ in $\overline{T}$, and leaves $\overline{T}$ unchanged.

It can be seen with a little thought that the fortified rollout algorithm will follow the initial trajectory $\overline{T}$, the one generated by the base heuristic starting from x_0, up to the point where it will discover a new feasible trajectory with smaller cost to replace $\overline{T}$. Similarly, the new trajectory $\overline{T}$ may be subsequently replaced by another feasible trajectory with smaller cost, etc. Note that if the base heuristic is sequentially improving, the fortified rollout algorithm generates the same trajectory as the (nonfortified) rollout algorithm given earlier. However, it can be verified, by modifying the proof of Props. 3.1 and 3.2, that even when the base heuristic is not sequentially improving, the fortified rollout algorithm will generate a trajectory that is feasible and has cost that is no worse than the cost of the trajectory generated by the base heuristic starting from x_0.

Extension for Time-Additive Constraints

In our formulation, we have assumed that the partial trajectories $R\big(f_k(\overline{x}_k, u_k)\big)$ generated by the base heuristic [cf. Eq. (2.2)] do not depend on the preceding states and controls

$$x_0, \overline{u}_0, \ldots, \overline{u}_{k-1}, \overline{x}_k, u_k.$$

On the other hand such dependence may be desirable to enhance the possibility that the trajectories

$$T_k^c(u_k) = (x_0, \overline{u}_0, \ldots, \overline{u}_{k-1}, \overline{x}_k) \cup (u_k) \cup H(x_{k+1})$$

satisfy the constraint $T_k^c(u_k) \in C$.

For example, consider the case where C is specified by the time-additive constraints

$$g_N^m(x_N) + \sum_{k=0}^{N-1} g_k^m(x_k, u_k) \leq b^m, \qquad m = 1, \ldots, M, \tag{4.1}$$

[cf. Eq. (1.3)], representing restrictions on M resources. Then, it is natural for the base heuristic to take into account the amounts of resources already expended. One way to do this is by augmenting the state x_k to include the quantities

$$y_k^m = \sum_{i=0}^{k-1} g_i^m(x_i, u_i), \qquad m = 1, \ldots, M, \ k = 1, \ldots, N-2,$$

$$y_0^m = 0, \qquad y_N^m = g_N^m\big(f_{N-1}(x_{N-1}, u_{N-1})\big) + \sum_{i=0}^{N-1} g_i^m(x_i, u_i).$$

In other words, we may reformulate the state of the system to be $(x_k, y_k^1, \ldots, y_k^M)$, and to evolve according to the new system equation

$$x_{k+1} = f_k(x_k, u_k), \qquad k = 0, \ldots, N-1,$$

$$y_{k+1}^m = y_k^m + g_k^m(x_k, u_k), \quad m = 1, \ldots, M, \ k = 0, \ldots, N-2,$$

$$y_0^m = 0, \qquad y_N^m = y_{N-1}^m + g_N^m\big(f_{N-1}(x_{N-1}, u_{N-1})\big) + g_{N-1}^m(x_{N-1}, u_{N-1}).$$

The time-additive constraints (4.1) are then reformulated as $y_N^m \leq b^m$, $m = 1, \ldots, M$. The rollout algorithm of Section 2, when used in conjunction with this reformulated problem, allows for base heuristics whose generated trajectories depend not only on x_k but also on $y_k^1, \ldots, y_k^M$, and brings the analysis of Section 3 to bear.

Tree-Based Rollout Algorithm

It is possible to improve the performance of the rollout algorithm at the expense of maintaining more than one partial trajectory. In particular, instead of the partial trajectory T_k of Eq. (2.1), we can maintain a *tree* of partial trajectories that start at x_0. These trajectories need not be of equal length, i.e., they need not have the same number of stages. At each step of the algorithm, we select a single partial trajectory from this tree. Let this partial trajectory have k stages and denote it by T_k. If $\overline{x}_k$ is the terminal state of T_k, we extend T_k by possibly more than one state subsequent to $\overline{x}_k$, similar to the rollout algorithm of Section 2.

This is done as follows: given $\overline{x}_k$, form the sets $T_k^c(u_k)$ and $\overline{U}_k(\overline{x}_k)$ of Eqs. (2.2) and (2.3), and select a subset $\hat{U}_k(x_k)$ of controls $\overline{u}_k \in \overline{U}_k(\overline{x}_k)$ for which the quantity

$$g_k(\overline{x}_k, u_k) + \tilde{J}\big(f_k(\overline{x}_k, u_k)\big) \tag{4.2}$$

is "small" according to some threshold criterion. The subset of controls $\overline{u}_k \in \hat{U}_k(\overline{x}_k)$ and the corresponding states $\overline{x}_{k+1} = f_k(\overline{x}_k, \overline{u}_k)$ are then added to the tree of trajectories mantained by the rollout algorithm. The algorithm terminates when all the trajectories of the tree it maintains are complete. Thus, the end result of the algorithm is a tree of (complete) trajectories that are feasible and start at the given initial state x_0. A trajectory from this tree that has minimum cost is then selected.

Note that as long as the set $\hat{U}_k(\overline{x}_k)$ contains a control that minimizes the expression (4.2) over $\overline{U}_k(\overline{x}_k)$, the cost improvement property of Section 3 still holds. However, the tree-based algorithm may have improved performance, essentially because it examines more alternative trajectories. Note also that there is considerable freedom for selecting the subset $\hat{U}_k(x_k)$, the size of which determines the computational cost of the algorithm.

We finally mention a drawback of the tree-based algorithm: it is suitable for off-line computation, but it cannot be applied in an on-line context, where the rollout control selection is made after the current state becomes known as the system evolves in real-time. By contrast, the rollout algorithm of Section 2 is well-suited for on-line applications.

5. REFERENCES

[BeC99] Bertsekas, D. P., and Castanon, D. A., 1999. "Rollout Algorithms for Stochastic Scheduling Problems," Heuristics, Vol. 5, pp. 89-108.

[BeD02] Bertsimas, D., and Demir, R., 2002. "An Approximate Dynamic Programming Approach to Multi-Dimensional Knapsack Problems," Management Science, Vol. 4, pp. 550-565.

[BeP03] Bertsimas, D., and Popescu, I., 2003. "Revenue Management in a Dynamic Network Environment,"Transportation Science, Vol. 37, pp. 257-277.

[BeT96] Bertsekas, D. P., and Tsitsiklis, J. N., 1996. Neuro-Dynamic Programming, Athena Scientific, Belmont, MA.

[Ber97] Bertsekas, D. P., 1997. "Differential Training of Rollout Policies," Proc. of the 35th Allerton Conference on Communication, Control, and Computing, Allerton Park, Ill.

[Ber01] Bertsekas, D. P., 2001. Dynamic Programming and Optimal Control, 2nd Edition, Athena Scientific, Belmont, MA.

[CGC04] Chang, H. S., Givan, R. L., and Chong, E. K. P., 2004. "Parallel Rollout for Online Solution of Partially Observable Markov Decision Processes," Discrete Event Dynamic Systems, Vol. 14, pp. 309-341.

[Chr97] Christodouleas, J. D., 1997. "Solution Methods for Multiprocessor Network Scheduling Problems with Application to Railroad Operations," Ph.D. Thesis, Operations Research Center, Massachusetts Institute of Technology.

[FeV02] Ferris, M. C., and Voelker, M. M., 2002. "Neuro-Dynamic Programming for Radiation Treatment Planning," Numerical Analysis Group Research Report NA-02/06, Oxford University Computing Laboratory, Oxford University.

[FeV04] Ferris, M. C., and Voelker, M. M., 2004. "Fractionation in Radiation Treatment Planning," Mathematical Programming B, Vol. 102, pp. 387-413.

[GuM01] Guerriero, F., and Musmanno, R., 2001. "Label Correcting Methods to Solve Multicriteria Shortest Path Problems," J. Optimization Theory Appl., Vol. 111, pp. 589-613.

[GuM03] Guerriero, F., and Mancini, M., 2003. "A Cooperative Parallel Rollout Algorithm for the Sequential Ordering Problem," Parallel Computing, Vol. 29, pp. 663-677.

[Jaf84] Jaffe, J. M., 1984. "Algorithms for Finding Paths with Multiple Constraints," Networks, Vol. 14, pp. 95-116.

[MMB02] McGovern, A., Moss, E., and Barto, A., 2002. "Building a Basic Building Block Scheduler Using Reinforcement Learning and Rollouts," Machine Learning, Vol. 49, pp. 141-160.

[Mar84] Martins, E. Q. V., 1984. "On a Multicriteria Shortest Path Problem," European J. of Operational Research, Vol. 16, pp. 236-245.

[MPP04] Meloni, C., Pacciarelli, D., and Pranzo, M., 2004. "A Rollout Metaheuristic for Job Shop Scheduling Problems," Annals of Operations Research, Vol. 131, pp. 215-235.

[SGC02] Savagaonkar, U., Givan, R., and Chong, E. K. P., 2002. "Sampling Techniques for Zero-Sum, Discounted Markov Games," in Proc. 40th Allerton Conference on Communication, Control and Computing, Monticello, Ill.

[Sec00] Secomandi, N., 2000. "Comparing Neuro-Dynamic Programming Algorithms for the Vehicle Routing Problem with Stochastic Demands," Computers and Operations Research, Vol. 27, pp. 1201-1225.

[Sec01] Secomandi, N., 2001. "A Rollout Policy for the Vehicle Routing Problem with Stochastic Demands," Operations Research, Vol. 49, pp. 796-802.

[Sec03] Secomandi, N., 2003. "Analysis of a Rollout Approach to Sequencing Problems with Stochastic Routing Applications," J. of Heuristics, Vol. 9, pp. 321-352.

[StW91] Stewart, B. S., and White, C. C., 1991. "Multiobjective A^*," J. ACM, Vol. 38, pp. 775-814.

[TPL04] Tu, F., Pham, D., Luo, J., Pattipati, K. R., and Willett, P., 2004. "Decision Feedback with Rollout for Multiuser Detection in Synchronous CDMA," IEE Proceedings-Communications, Vol. 151,pp. 383-386.

[TeG96] Tesauro, G., and Galperin, G. R., 1996. "On-Line Policy Improvement Using Monte Carlo Search," presented at the 1996 Neural Information Processing Systems Conference, Denver, CO; also in M. Mozer et al. (eds.), Advances in Neural Information Processing Systems 9, MIT Press (1997).

[TuP03] Tu, F., and Pattipati, K. R., 2003. "Rollout Strategies for Sequential Fault Diagnosis," IEEE Trans. on Systems, Man and Cybernetics, Part A, pp. 86-99.

[WCG03] Wu, G., Chong, E. K. P., and Givan, R. L., 2003. "Congestion Control Using Policy Rollout," Proc. 2nd IEEE CDC, Maui, Hawaii, pp. 4825-4830.

[YDR05] Yan, X., Diaconis, P., Rusmevichientong, P., and Van Roy, B., 2005. "Solitaire: Man Versus Machine," Advances in Neural Information Processing Systems, Vol. 17, to appear.