European Journal of Control (2005)11:310–334
© 2005 EUCA

European Journal of Control

Dynamic Programming and Suboptimal Control: A Survey from ADP to MPC*

Dimitri P. Bertsekas**

Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA 02139, USA

We survey some recent research directions within the field of approximate dynamic programming, with a particular emphasis on rollout algorithms and model predictive control (MPC). We argue that while they are motivated by different concerns, these two methodologies are closely connected, and the mathematical essence of their desirable properties (cost improvement and stability, respectively) is couched on the central dynamic programming idea of policy iteration. In particular, among other things, we show that the most common MPC schemes can be viewed as rollout algorithms and are related to policy iteration methods. Furthermore, we embed rollout and MPC within a new unifying suboptimal control framework, based on a concept of restricted or constrained structure policies, which contains these schemes as special cases.

Keywords: dynamic programming, stochastic optimal control, model predictive control, rollout algorithm

1. Introduction

We consider a basic stochastic optimal control problem, which is amenable to a dynamic programming solution, and is considered in many sources (including the author's dynamic programming textbook [14], whose notation we adopt). We have a discrete-time dynamic system

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, 1, \ldots, \tag{1.1}$$

where x_k is the state taking values in some set, u_k is the control to be selected from a finite set $U_k(x_k)$, w_k is a random disturbance and f_k is a given function. We assume that each w_k is selected according to a probability distribution that may depend on x_k and u_k, but not on previous disturbances. The cost incurred at the k-th time period is denoted by $g_k(x_k, u_k, w_k)$. For mathematical rigor (to avoid measurability questions), we assume that w_k takes values in a countable set, but the following discussion applies qualitatively to more general cases.

We initially assume perfect state information, i.e. that the control u_k is chosen with knowledge of the current state x_k; we later discuss the case of imperfect state information, where u_k is chosen with knowledge of a history of measurements related to the state. We, thus, initially consider policies $\pi = \{\mu_0, \mu_1, \ldots\}$, which at time k map a state x_k to a control $\mu_k(x_k) \in U_k(x_k)$. We focus primarily on an N-stage horizon problem where k takes the values $0, 1, \ldots, N-1$, and there is a terminal cost $g_N(x_N)$ that depends on the terminal state x_N. The cost-to-go of π starting from a state x_k at time k is denoted by

$$J_k^\pi(x_k) = E\left\{ g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, \mu_i(x_i), w_i) \right\}. \tag{1.2}$$

The optimal cost-to-go starting from a state x_k at time k is

$$J_k(x_k) = \inf_\pi J_k^\pi(x_k),$$

*Many thanks are due to Janey Yu for helpful comments. Research supported by NSF Grant ECS-0218328.
**E-mail: dimitrib@mit.edu

Received 15 June 2005; Accepted 30 June 2005.
Recommended by E.F. Camacho, R. Tempo, S. Yurkovich, P.J. Fleming

and it is assumed that $J_k^\pi(x_k)$ and $J_k(x_k)$ are finite for all x_k, π and k. The cost-to-go functions J_k satisfy the following recursion of dynamic programming (DP)

$$J_k(x_k) = \inf_{u_k \in U_k(x_k)} E\{g_k(x_k, u_k, w_k) + J_{k+1} \times (f_k(x_k, u_k, w_k))\}, \quad k = 0, 1, \ldots, N-1, \tag{1.3}$$

with the initial condition

$$J_N(x_N) = g_N(x_N).$$

Our discussion applies with minor modifications to infinite horizon problems, with the DP algorithm replaced by its asymptotic form (Bellman's equation). For example, for a stationary discounted cost problem, the analog of the DP algorithm (1.3) is

$$J(x) = \inf_{u \in U(x)} E\{g(x, u, w) + \alpha J(f(x, u, w))\}, \quad \forall x, \tag{1.4}$$

where $J(x)$ is the optimal (α-discounted) cost-to-go starting from x.

An optimal policy may be obtained in principle by minimization in the right-hand side of the DP algorithm (1.3), but this requires the calculation of the optimal cost-to-go functions J_k, which for many problems is prohibitively time-consuming. This has motivated approximations that require a more tractable computation, but yield a suboptimal policy. There is a long history of such suboptimal control methods, and the purpose of this paper is to survey some of them, to discuss their connections, and to place them under the umbrella of a unified methodology. Although we initially assume a stochastic model with perfect state information, much of the subsequent material is focused on other types of models, including deterministic models.

A broad class of suboptimal control methods, which we refer to as *approximate dynamic programming* (ADP), is based on replacing the cost-to-go function J_{k+1} in the right-hand side of the DP algorithm (1.3) by an approximation $\tilde{J}_{k+1}$, with $\tilde{J}_N = g_N$. Thus, this method applies at time k and state x_k a control $\overline{\mu}_k(x_k)$ that minimizes over $u_k \in U_k(x_k)$

$$E\{g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k))\}.$$

The corresponding suboptimal policy $\overline{\pi} = \{\overline{\mu}_0, \overline{\mu}_1, \ldots, \overline{\mu}_{N-1}\}$ is determined by the approximate cost-to-go functions $\tilde{J}_1, \tilde{J}_2, \ldots, \tilde{J}_N$ (given either by their functional form or by an algorithm to calculate their values at states of interest). Note that if the problem is stationary, and the functions $\tilde{J}_{k+1}$ are all equal, as they would normally be in an infinite horizon context, the policy $\overline{\pi}$ is stationary.

There are several alternative approaches for selecting or calculating the functions $\tilde{J}_{k+1}$. We distinguish two broad categories:

(1) *Explicit cost-to-go approximation.* Here $\tilde{J}_{k+1}$ is computed *off-line* in one of a number of ways. Some important examples are as follows:
   (a) By solving (optimally) a related simpler problem, obtained for example by state aggregation or by some other type of problem simplification, such as some form of enforced decomposition. The functions $\tilde{J}_{k+1}$ are derived from the optimal cost-to-go functions of the simpler problem. We will not discuss this approach further in this paper, and we refer to the author's textbook [13] for more details.
   (b) By introducing a parametric approximation architecture, such as a neural network or a weighted sum of basis functions or features. The idea here is to approximate the optimal cost-to-go $J_{k+1}(x)$ with a function of a given parametric form $\tilde{J}_{k+1}(x) = \hat{J}_{k+1}(x, r_{k+1})$, where r_{k+1} is a parameter vector. This vector is tuned by some form of *ad hoc*/heuristic method (as for example in computer chess) or some systematic method (for example, of the type provided by the neuro-dynamic programming and reinforcement learning methodologies, such as temporal difference and Q-learning methods; see the monographs by Bertsekas and Tsitsiklis [8], and Sutton and Barto [SuB98], and the recent edited volume by Barto et al. [2], which contain extensive bibliographies). There has also been considerable recent related work based on linear programming, actor-critic, policy gradient and other methods (for a sampling of recent work, see de Farias and Van Roy [18,19], Konda [30], Konda and Tsitsiklis [29], Marbach and Tsitsiklis [36], and Rantzer [41], which contain many other references). Again, this approach will not be discussed further in this paper.

(2) *Implicit cost-to-go approximation.* Here the values of $\tilde{J}_{k+1}$ at the states $f_k(x_k, u_k, w_k)$ are computed *on-line* as needed, via some computation of future costs, starting from these states (optimal or suboptimal/heuristic, with or without a rolling

horizon). We will focus on a few possibilities, which we will discuss in detail in Sections 4 and 5:

(a) *Rollout*, where the cost-to-go of a suboptimal/heuristic policy (called the *base policy*) is used as $\tilde{J}_{k+1}$. This cost is computed as necessary in whatever form is most convenient, including by on-line simulation. It can be seen that the suboptimal policy $\overline{\pi}$ obtained by rollout is identical to the policy obtained by a *single policy improvement step* of the classical policy iteration method, starting from the base policy. There is also a variant of the rollout algorithm where $\tilde{J}_{k+1}$ is defined via a base policy but may differ from the cost-to-go of the corresponding policy. This variant still has the cost improvement property of policy iteration under some assumptions that are often satisfied (see Example 3.1 and Section 4).

(b) *Open-loop feedback control* (OLFC), where an optimal open-loop computation is used, starting from the state x_k (in the case of perfect state information) or the conditional probability distribution of the state (in the case of imperfect state information).

(c) *Model predictive control* (MPC), where an optimal control computation is used in conjunction with a rolling horizon. This computation is deterministic, possibly based on a simplification of the original problem via certainty equivalence, but there is also a minimax variant that implicitly involves reachability of target tube computations (see Section 5).

A few important variations of the preceding schemes should be mentioned. The first is the use of *multistep lookahead*, which aims to improve the performance of one-step lookahead, at the expense of increased on-line computation. The second is the use of *certainty equivalence*, which simplifies the off-line and on-line computations by replacing the current and future unknown disturbances $w_k, \ldots, w_{N-1}$ with nominal deterministic values. A third variation, which applies to *problems of imperfect state information*, is to use one of the preceding schemes with the unknown state x_k replaced by some estimate. We briefly discuss some of these variations in the next section.

Our paper has three objectives:

(1) To derive some general performance bounds, and relate them to observed practical performance, and to the issue of stability in MPC. This is done in Section 3, and also in Section 5, where MPC is discussed in more detail.

(2) To demonstrate the connection between rollout (and hence policy iteration) on the one hand, and OLFC and MPC on the other hand. Indeed, we will show that both OLFC and MPC are special cases of rollout, obtained for particular choices of the corresponding base policy. This is explained in Sections 4 and 5.

(3) To introduce a general unifying framework for suboptimal control, which includes as special cases rollout, OLFC, and MPC, and captures the mathematical essence of their attractive properties. This is the subject of Section 6.

A complete bibliography is beyond the scope of the present paper. In this section, we will selectively provide a few major relevant sources, with preference given to survey papers and textbooks. We supplement these references with more specific remarks at the appropriate points in subsequent sections.

The main idea of rollout algorithms, obtaining an improved policy starting from some other suboptimal policy using a one-time policy improvement, has appeared in several DP application contexts, although the connection with policy iteration has not always been recognized. In the context of game-playing computer programs, it has been proposed by Abramson [1] and by Tesauro [TeG96]. The name 'rollout' was coined by Tesauro in specific reference to rolling the dice in the game of backgammon. In Tesauro's proposal, a given backgammon position is evaluated by 'rolling out' many games starting from that position, using a simulator, and the results are averaged to provide a 'score' for the position. The internet contains a lot of material on computer backgammon and the use of rollout, in some cases in conjunction with multistep lookahead and cost-to-go approximation.

Generally, it is possible to use a base policy to compute cost-to-go approximations $\tilde{J}_k$, but to make a strong connection with policy iteration, it is essential that $\tilde{J}_k$ be the true cost-to-go function of some policy. For deterministic problems, this is equivalent to requiring that the base policy has a property called *sequential consistency*. However, for cost improvement it is sufficient that the base policy has a weaker property, called *sequential improvement*, which bears a relation to concepts of Lyapounov stability theory and can also be extended to general limited lookahead policies (see Proposition 3.1 and Example 3.1). The sequential consistency and sequential improvement properties for deterministic problems were first formalized by Bertsekas et al. [3], and will be described in Section 4 in a more general context, which includes constraints.

For further work on rollout algorithms, see Christodouleas [Chr97], Bertsekas [12], Bertsekas and Castanon [4], Secomandi [43–45], Bertsimas and Demir [5], Ferris and Voelker [23,24], McGovern et al. [31], Savagaonkar et al. [SGC02], Bertsimas and Popescu [6], Guerriero and Mancini [GuM03], Tu and Pattipati [47], Wu et al. [48], Chang et al. [16], Meloni et al. [32] and Yan et al. [52]. Collectively, these works discuss a broad variety of applications and case studies, and generally report positive computational experience.

The MPC approach has become popular in a variety of control system design contexts, and particularly in chemical process control, where meeting explicit control and state constraints is an important practical issue. Over time, there has been increasing awareness of the connection with the problem of reachability of target tubes, set-membership descriptions of uncertainty, and minimax control (see the discussion of Section 5). The associated literature is voluminous and we make no attempt to provide detailed references. The stability analysis given here is based on the work of Keerthi and Gilbert [28]. For extensive surveys of the field, which give many references, see Morari and Lee [39], Mayne et al. [33], Rawlings [42], Findeisen et al. [22] and Qin and Badgwell [40]. For related textbooks, see Martin-Sanchez and Rodellar [34], Maciejowski [35] and Camacho and Bordons [17].

While our main emphasis in this paper is on highlighting the conceptual connections between several approaches to suboptimal control, we also present a few new and/or unpublished results. In particular, the material of Section 4 on rollout algorithms for constrained DP is unpublished (it is also reported separately in Ref. [14]). The connection of MPC with rollout algorithms reported in Section 5 does not seem to have been noticed earlier. Finally, the material of Section 6 on the unifying suboptimal control framework based on restricted structure policies is new.

2. Limited Lookahead Policies

An effective way to reduce the computation required by DP is to truncate the time horizon, and use at each stage a decision based on lookahead of a small number of stages. The simplest possibility is to use a *one-step lookahead policy* whereby at stage k and state x_k one uses a control $\overline{\mu}_k(x_k)$, which attains the minimum in the expression

$$\min_{u_k \in U_k(x_k)} E\{g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k))\}, \tag{2.1}$$

where $\tilde{J}_{k+1}$ is some approximation of the true cost-to-go function J_{k+1}, with $\tilde{J}_N = g_N$. Similarly, a *two-step lookahead policy* applies at time k and state x_k, the control $\overline{\mu}_k(x_k)$ attaining the minimum in the preceding equation, where now $\tilde{J}_{k+1}$ is obtained itself on the basis of a one-step lookahead approximation. In other words, for all possible states x_{k+1} that can be generated via the system equation starting from x_k,

$$x_{k+1} = f_k(x_k, u_k, w_k),$$

we have

$$\tilde{J}_{k+1}(x_{k+1}) = \min_{u_{k+1} \in U_{k+1}(x_{k+1})} E\{g_{k+1}(x_{k+1}, u_{k+1}, w_{k+1}) + \tilde{J}_{k+2}(f_{k+1}(x_{k+1}, u_{k+1}, w_{k+1}))\},$$

where $\tilde{J}_{k+2}$ is some approximation of the cost-to-go function J_{k+2}. Policies with lookahead of more than two stages are similarly defined.

Note that even with readily available cost-to-go approximations $\tilde{J}_k$, the minimization over $u_k \in U_k(x_k)$ in the calculation of the one-step lookahead control [cf. Eq. (2.1)] may involve substantial computation. This motivates several variants/simplifications of the basic method. For example, the minimization over $U_k(x_k)$ in Eq. (2.1) may be replaced by a minimization over a subset

$$\overline{U}_k(x_k) \subset U_k(x_k).$$

Thus, the control $\overline{\mu}_k(x_k)$ used in this variant is one that attains the minimum in the expression

$$\min_{u_k \in \overline{U}_k(x_k)} E\{g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k))\}. \tag{2.2}$$

A practical example of this approach is when by using some heuristic or approximate optimization, we identify a subset $\overline{U}_k(x_k)$ of promising controls, and to save computation, we restrict attention to this subset in the one-step lookahead minimization.

Another major simplification is common in problems of imperfect state information, where the computational requirements of exact DP are often prohibitive. There, the state x_k is not known exactly but may be estimated based on observations obtained up to time k. A possible approach is then to replace the state x_k in Eqs (2.1) and (2.2) by an estimate.

A third simplification, known as (assumed) *certainty equivalence*, aims to avoid the computation of the expected value in Eqs (2.1) and (2.2), and to simplify the calculation of the required values of $\tilde{J}_{k+1}$ by replacing the stochastic quantities w_k with some

nominal deterministic values $\overline{w}_k$. Then, the one-step-lookahead minimization in Eq. (2.1) is replaced by

$$\min_{u_k \in U_k(x_k)} [g_k(x_k, u_k, \overline{w}_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, \overline{w}_k))]. \tag{2.3}$$

The cost-to-go approximation $\tilde{J}_{k+1}$ is often obtained by solving an optimal control problem where the future uncertain quantities $w_{k+1}, \ldots, w_{N-1}$ are replaced by deterministic nominal values $\overline{w}_{k+1}, \ldots, \overline{w}_{N-1}$. Then, the minimization in Eq. (2.3) can be done by solving a deterministic optimal control problem, from the present time to the end of the horizon:

(1) Find a control sequence $\{\overline{u}_k, \overline{u}_{k+1}, \ldots, \overline{u}_{N-1}\}$ that minimizes

$$g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, u_i, \overline{w}_i)$$

subject to the constraints

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

(2) Use as control the first element in the control sequence found:

$$\overline{\mu}_k(x_k) = \overline{u}_k.$$

Note that in variants of Step (1) above, the deterministic optimization problem may still be difficult, and may be solved approximately using a suboptimal method that offers some computational advantage. For example, when the number of stages N is large or infinite, the deterministic problem may be defined over a rolling horizon with a fixed number of stages. We finally note that certainty equivalence is often used at the modeling level, when a deterministic model is adopted for a system which is known to involve uncertainty.

3. Error Bounds

For any suboptimal control scheme it is important to have theoretical bounds on its performance. Unfortunately, despite recent progress, the methodology for performance analysis of suboptimal control schemes is not very satisfactory at present, and the validation of a suboptimal policy by simulation is often essential in practice. Still, however, with experience and new theoretical research, the relative merits of different approaches have been clarified to some extent, and it is now understood that some schemes possess desirable theoretical performance guarantees, while others do not. In this section, we will discuss a few performance bounds that are available for ADP. For additional theoretical analysis on performance bounds, based on different approaches, see Witsenhausen [50,51].

Let us denote by $\overline{J}_k(x_k)$ the expected cost-to-go incurred by a limited lookahead policy $\{\overline{\mu}_0, \overline{\mu}_1, \ldots, \overline{\mu}_{N-1}\}$ starting from state x_k at time k [$\overline{J}_k(x_k)$ should be distinguished from $\tilde{J}_k(x_k)$, the approximation of the cost-to-go that is used to compute the limited lookahead policy via the minimization in Eq. (2.1) or Eq. (2.2)]. It is generally difficult to evaluate analytically the functions $\overline{J}_k$, even when the functions $\tilde{J}_k$ are readily available. We, thus, aim to obtain some estimates of $\overline{J}_k(x_k)$. The following proposition gives a condition under which the one-step lookahead policy achieves a cost $\overline{J}_k(x_k)$ which is better than the approximation $\tilde{J}_k(x_k)$. The proposition also provides a readily computable upper bound to $\overline{J}_k(x_k)$.

Proposition 3.1. Assume that for all x_k and k, we have

$$\min_{u_k \in \overline{U}_k(x_k)} E\{g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k))\} \leq \tilde{J}_k(x_k). \tag{3.1}$$

Then the cost-to-go functions $\overline{J}_k$ corresponding to a one-step lookahead policy that uses $\tilde{J}_k$ and $\overline{U}_k(x_k)$ with $\overline{U}_k(x_k) \subset U_k(x_k)$ [cf. Eq. (2.2)] satisfy for all x_k and k

$$\overline{J}_k(x_k) \leq \min_{u_k \in \overline{U}_k(x_k)} E\{g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k))\}. \tag{3.2}$$

Proof. For $k = 0, \ldots, N-1$, denote

$$\hat{J}_k(x_k) = \min_{u_k \in \overline{U}_k(x_k)} E\{g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k))\}, \tag{3.3}$$

and let $\hat{J}_N = g_N$. We must show that for all x_k and k, we have $\overline{J}_k(x_k) \leq \hat{J}_k(x_k)$. We use backwards induction on k. In particular, we have $\overline{J}_N(x_N) = \hat{J}_N(x_N) = \tilde{J}_N(x_N) = g_N(x_N)$ for all x_N. Assuming that $\overline{J}_{k+1}(x_{k+1}) \leq \hat{J}_{k+1}(x_{k+1})$ for all x_{k+1}, we have

$$\begin{aligned}
\overline{J}_k(x_k) &= E\{g_k(x_k, \overline{\mu}_k(x_k), w_k) + \overline{J}_{k+1}(f_k(x_k, \overline{\mu}_k(x_k), w_k))\} \\
&\leq E\{g(x_k, \overline{\mu}_k(x_k), w_k) + \hat{J}_{k+1}(f_k(x_k, \overline{\mu}_k(x_k), w_k))\} \\
&\leq E\{g(x_k, \overline{\mu}_k(x_k), w_k) + \tilde{J}_{k+1}(f_k(x_k, \overline{\mu}_k(x_k), w_k))\} \\
&= \hat{J}_k(x_k),
\end{aligned}$$

for all x_k. The first equality above follows from the DP algorithm that defines the costs-to-go $\overline{J}_k$ of the limited lookahead policy, while the first inequality follows from the induction hypothesis, and the second inequality follows from the assumption (3.1). This completes the induction proof. □

Note that by Eq. (3.2), the value $\hat{J}_k(x_k)$ of Eq. (3.3), which is the calculated one-step lookahead cost from state x_k at time k, provides a readily obtainable performance bound for the cost-to-go $\overline{J}_k(x_k)$ of the one-step lookahead policy. Furthermore, using also the assumption (3.1), we obtain for all x_k and k,

$$\overline{J}_k(x_k) \leq \tilde{J}_k(x_k),$$

that is, *the cost-to-go of the one-step lookahead policy is no greater than the lookahead approximation on which it is based.* The critical assumption (3.1) in Proposition 3.1 can be verified in a few interesting special cases, as indicated by the following examples.

Example 3.1 (*Rollout algorithm*). Suppose that $\tilde{J}_k(x_k)$ is the cost-to-go $J_k^{\pi}(x_k)$ of some given suboptimal policy $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$ and that the set $\overline{U}_k(x_k)$ contains the control $\mu_k(x_k)$ for all x_k and k. The resulting one-step lookahead algorithm is called the *rollout algorithm* and will be discussed further in Section 4. From the DP algorithm (restricted to the given policy π), we have

$$\tilde{J}_k(x_k) = E\{g_k(x_k, \mu_k(x_k), w_k) + \tilde{J}_{k+1}(f_k(x_k, \mu_k(x_k), w_k))\},$$

which in view of the assumption $\mu_k(x_k) \in \overline{U}_k(x_k)$, yields

$$\tilde{J}_k(x_k) \geq \min_{u_k \in \overline{U}_k(x_k)} E\{g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k))\}. \tag{3.4}$$

Thus, the assumption of Proposition 3.1 is satisfied, and it follows that the rollout algorithm performs no worse than the policy on which it is based, starting from any state and stage.

In a generalization of the rollout algorithm, which is relevant to the algorithm for constrained DP given in Section 4, and to MPC as described in Section 5, $\tilde{J}_k(x_k)$ is a lower bound to the cost-to-go of some given suboptimal policy $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, starting from x_k, and furthermore satisfies the inequality

$$\tilde{J}_k(x_k) \geq E\{g_k(x_k, \mu_k(x_k), w_k) + \tilde{J}_{k+1}(f_k(x_k, \mu_k(x_k), w_k))\},$$

for all x_k. Then, Eq. (3.4) holds, and from Proposition 3.1, it follows that $\overline{J}_k(x_k) \leq \tilde{J}_k(x_k)$. Since $\tilde{J}_k(x_k) \leq J_k^{\pi}(x_k)$ by assumption, we see that again the rollout algorithm performs no worse than the policy on which it is based, starting from any state and stage.

Example 3.2 (*Rollout algorithm with multiple heuristics*). Consider a scheme that is similar to the one of the preceding example, except that $\tilde{J}_k(x_k)$ is the minimum of the cost-to-go functions corresponding to m heuristics, i.e.

$$\tilde{J}_k(x_k) = \min\{J_k^{\pi^1}(x_k), \ldots, J_k^{\pi^m}(x_k)\},$$

where for each j, $J_k^{\pi^j}(x_k)$ is the cost-to-go of a policy $\pi^j = \{\mu_{j,0}, \ldots, \mu_{j,N-1}\}$, starting from state x_k at stage k. From the DP algorithm, we have, for all j,

$$J_k^{\pi^j}(x_k) = E\{g_k(x_k, \mu_{j,k}(x_k), w_k) + J_{k+1}^{\pi^j}(f_k(x_k, \mu_{j,k}(x_k), w_k))\},$$

from which, using the definition of $\tilde{J}_k$, it follows that

$$\begin{aligned} J_k^{\pi^j}(x_k) &\geq E\{g_k(x_k, \mu_{j,k}(x_k), w_k) + \tilde{J}_{k+1}(f_k(x_k, \mu_{j,k}(x_k), w_k))\} \\ &\geq \min_{u_k \in \overline{U}_k(x_k)} E\{g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k))\}. \end{aligned}$$

Taking the minimum of the left-hand side over j, we obtain

$$\tilde{J}_k(x_k) \geq \min_{u_k \in \overline{U}_k(x_k)} E\{g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k))\}.$$

Thus, Proposition 3.1 implies that the one-step lookahead algorithm based on the heuristic algorithms' costs-to-go $J_k^{\pi^1}(x_k), \ldots, J_k^{\pi^m}(x_k)$ performs better than *all* of these heuristics, starting from any state and stage. This also follows from the analysis of the preceding example, since $\tilde{J}_k(x_k)$ is a lower bound to all the heuristic costs-to-go $J_k^{\pi^j}(x_k)$.

Generally, the approximate cost-to-go functions $\tilde{J}_k$ need not satisfy the assumption (3.1) of Proposition 3.1. The following proposition does not require this assumption, and applies to any policy, not necessarily one obtained by one-step or multi-step lookahead. It is useful in some contexts, including the case where the minimization involved in the calculation in the one-step lookahead policy is not exact.

Proposition 3.2. Let $\tilde{J}_k, k = 0, 1, \ldots, N$, be functions of x_k, with $\tilde{J}_N(x_N) = g_N(x_N)$ for all x_N. Let also

$\pi = \{\overline{\mu}_0, \overline{\mu}_1, \ldots, \overline{\mu}_{N-1}\}$ be a policy such that for all x_k and k, we have

$$\begin{aligned}\tilde{J}_k(x_k) + \gamma_k &\le E\{g_k(x_k, \overline{\mu}_k(x_k), w_k) \\ &\quad + \tilde{J}_{k+1}(f_k(x_k, \overline{\mu}_k(x_k), w_k))\} \\ &\le \tilde{J}_k(x_k) + \delta_k, \end{aligned} \tag{3.5}$$

where $\gamma_k, \delta_k, k = 0, \ldots, N-1$, are some scalars. Then for all x_k and k, we have

$$\tilde{J}_k(x_k) + \sum_{i=k}^{N-1} \gamma_i \le J_k^{\pi}(x_k) \le \tilde{J}_k(x_k) + \sum_{i=k}^{N-1} \delta_i, \tag{3.6}$$

where $J_k^{\pi}(x_k)$ is the cost-to-go of π starting from state x_k at stage k.

Proof. We use backwards induction on k to prove the right-hand side of Eq. (3.5). The proof of the left-hand side is similar. In particular, we have $J_N^{\pi}(x_N) = \tilde{J}_N(x_N) = g_N(x_N)$ for all x_N. Assuming that

$$J_{k+1}^{\pi}(x_{k+1}) \le \tilde{J}_{k+1}(x_{k+1}) + \sum_{i=k+1}^{N-1} \delta_i$$

for all x_{k+1}, we have

$$\begin{aligned} J_k^{\pi}(x_k) &= E\{g_k(x_k, \overline{\mu}_k(x_k), w_k) \\ &\quad + J_{k+1}^{\pi}(f_k(x_k, \overline{\mu}_k(x_k), w_k))\} \\ &\le E\{g(x_k, \overline{\mu}_k(x_k), w_k) \\ &\quad + \tilde{J}_{k+1}(f_k(x_k, \overline{\mu}_k(x_k), w_k))\} + \sum_{i=k+1}^{N-1} \delta_i \\ &\le \tilde{J}_k(x_k) + \delta_k + \sum_{i=k+1}^{N-1} \delta_i, \end{aligned}$$

for all x_k. The first equality above follows from the DP algorithm that defines the costs-to-go J_k^{π} of π, while the first inequality follows from the induction hypothesis, and the second inequality follows from the assumption (3.6). This completes the induction proof. □

Example 3.3 (*Certainty equivalent control*). Consider the certainty equivalent control scheme (CEC), where each disturbance w_k is fixed at a nominal value $\overline{w}_k, k = 0, \ldots, N-1$, which is independent of x_k and u_k. Let $\tilde{J}_k(x_k)$ be the optimal value of the problem solved by CEC at state x_k and stage k:

$$\tilde{J}_k(x_k) = \min_{\substack{x_{i+1} = f_i(x_i, u_i, \overline{w}_i) \\ u_i \in U_i(x_i), i = k, \ldots, N-1}} \left[g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, u_i, \overline{w}_i) \right],$$

and let $\tilde{J}_N(x_N) = g_N(x_N)$ for all x_N. Recall that the CEC applies the control $\overline{\mu}_k(x_k) = \overline{u}_k$ after finding an optimal control sequence $\{\overline{u}_k, \ldots, \overline{u}_{N-1}\}$ for the deterministic problem in the right-hand side above. Note also that the following DP equation

$$\begin{aligned}\tilde{J}_k(x_k) = \min_{u_k \in U_k(x_k)} &[g_k(x_k, u_k, \overline{w}_k) \\ &+ \tilde{J}_{k+1}(f_k(x_k, u_k, \overline{w}_k))]\end{aligned}$$

holds, and that the control $\overline{u}_k$ applied by CEC minimizes in the right-hand side.

Let us now apply Proposition 3.2 to derive a performance bound for the CEC. We have for all x_k and k,

$$\begin{aligned}\tilde{J}_k(x_k) &= g_k(x_k, \overline{\mu}_k(x_k), \overline{w}_k) \\ &\quad + \tilde{J}_{k+1}(f_k(x_k, \overline{\mu}_k(x_k), \overline{w}_k)) \\ &= E\{g(x_k, \overline{\mu}_k(x_k), w_k) \\ &\quad + \tilde{J}_{k+1}(f_k(x_k, \overline{\mu}_k(x_k), w_k))\} - \gamma_k(x_k)\end{aligned}$$

where $\gamma_k(x_k)$ is defined by

$$\begin{aligned}\gamma_k(x_k) &= E\{g(x_k, \overline{\mu}_k(x_k), w_k) \\ &\quad + \tilde{J}_{k+1}(f_k(x_k, \overline{\mu}_k(x_k), w_k))\} \\ &\quad - g_k(x_k, \overline{\mu}_k(x_k), \overline{w}_k) \\ &\quad - \tilde{J}_{k+1}(f_k(x_k, \overline{\mu}_k(x_k), \overline{w}_k)).\end{aligned}$$

It follows that

$$\begin{aligned}&E\{g(x_k, \bar{\mu}_k(x_k), w_k) + \tilde{J}_{k+1}(f_k(x_k, \bar{\mu}_k(x_k), w_k))\} \\ &\quad \le \tilde{J}_k(x_k) + \delta_k,\end{aligned}$$

where

$$\delta_k = \max_{x_k} \gamma_k(x_k),$$

and by Proposition 3.2, we obtain the following bound for the cost-to-go function $\bar{J}_k(x_k)$ of the CEC:

$$\bar{J}_k(x_k) \le \tilde{J}_k(x_k) + \sum_{i=k}^{N-1} \delta_i.$$

The preceding performance bound is helpful when it can be shown that $\delta_k \le 0$ for all k, in which case we have $\bar{J}_k(x_k) \le \tilde{J}_k(x_k)$ for all x_k and k. This is true for example if for all x_k and u_k, we have

$$E\{g(x_k, u_k, w_k)\} \le g_k(x_k, u_k, \overline{w}_k),$$

and

$$E\{\tilde{J}_{k+1}(f_k(x_k, u_k, w_k))\} \le \tilde{J}_{k+1}(f_k(x_k, u_k, \overline{w}_k)).$$

The most common way to assert that inequalities of this type hold is via some kind of concavity assumptions; for example, the inequalities hold if the state, control, and disturbance spaces are Euclidean spaces, $\overline{w}_k$ is the expected value of w_k, and the functions $g(x_k, u_k, \cdot)$ and $\tilde{J}_{k+1}(f_k(x_k, u_k, \cdot))$, viewed as functions of w_k, are concave (this is Jensen's inequality, and follows easily from the definition of concavity). It can be shown that the concavity conditions just described are guaranteed if the system is linear with respect to x_k and w_k, the cost functions g_k are concave with respect to x_k and w_k for each fixed u_k, the terminal cost function g_N is concave, and the control constraint sets U_k do not depend on x_k.

4. Rollout and Open-Loop Feedback Control

In this section, we discuss rollout and OLFC schemes for one-step lookahead, and describe their relation. We first consider general stochastic problems, and we subsequently focus on deterministic problems, for which stronger results and algorithmic procedures are possible.

4.1. Rollout Algorithm

Consider the one-step lookahead minimization

$$\min_{u_k \in U_k(x_k)} E\{g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k))\}, \tag{4.1}$$

for the case where the approximating function $\tilde{J}_{k+1}$ is the cost-to-go J^{π}_{k+1} of some known heuristic/suboptimal policy $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, called *base policy* (see also Example 3.1). The policy thus obtained is called the *rollout policy* based on π. Thus *the rollout policy is a one-step lookahead policy, with the optimal cost-to-go approximated by the cost-to-go of the base policy*.

The salient feature of the rollout algorithm is its *cost improvement property*: it improves on the performance of the base policy (see Example 3.1). We note that the process of starting from some suboptimal policy and generating another policy using the one-step lookahead process described above is known as *policy improvement*, and is the basis of the *policy iteration* method, a primary method for solving DP problems. The rollout algorithm can be viewed as a single policy iteration, and its cost improvement property is a manifestation of the corresponding property of the policy iteration method. In practice, the rollout algorithm often performs surprisingly well (much better than its corresponding base policy). This is consistent with the observed practical behavior of the policy iteration method, which tends to produce large improvements in the first few iterations.

The necessary values of the approximate cost-to-go $\tilde{J}_{k+1}$ in Eq. (4.1) may be computed in a number of ways:

(1) By a closed-form expression. This may be possible in exceptional cases where the structure of the base policy is well-suited for a quasi-analytical cost-to-go evaluation.
(2) By an approximate off-line computation. For example, $\tilde{J}_{k+1}$ may be computed approximately as the output of a neural network or some other approximation architecture, which is trained by simulation.
(3) By an on-line computation, such as for example a form of simulation or optimization. The drawback of simulation is that its computational requirements may be excessive. Note, however, that for a deterministic problem, only one simulation trajectory is needed to calculate the cost-to-go of the base policy starting from some state, so in this case the computational requirements are greatly reduced.

A fuller discussion of the computational issues regarding the rollout algorithm is beyond the scope of this paper, and we refer to the literature on the subject. In particular, the textbook [13] contains an extensive account.

4.2. Rollout Algorithms for Deterministic Constrained DP

We now specialize the rollout algorithm to deterministic optimal control problems, and we also generalize it and extend its range of application to problems with some additional constraints, taking advantage of the special deterministic structure.

We consider a problem involving the system

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

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. The initial state is given and is denoted by x_0. Each control u_k must be chosen from a 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

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

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

$$V(T) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k), \tag{4.2}$$

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

$$T \in C, \tag{4.3}$$

where C is a given set of trajectories.

An optimal solution of this problem can be found by an extension of the DP algorithm, 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

$$g_N^m(x_N) + \sum_{k=0}^{N-1} g_k^m(x_k, u_k) \leq b^m, \quad m = 1, \ldots, M, \tag{4.4}$$

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 of suboptimal solution methods for the case where the constraints are of the form (4.4). Most of these proposals are cast in the context of the constrained and multiobjective shortest path problems; see, e.g. Jaffe [26], Martins [37], Guerriero and Musmanno [25] and Stewart and White [46], who also survey earlier work.

The extension of the rollout algorithm to deterministic problems with the constraints (4.3), uses a base policy, 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

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

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 policy 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 policy starting from the given initial state x_0 is feasible*, that is

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

Finding a base policy with this property may not be easy, but this question is beyond the scope this paper. It appears, however, that for many problems of interest, there are natural base policies that satisfy this feasibility requirement; this is true in particular when C is specified by Eq. (4.4) with only one constraint function ($M = 1$), in which case a feasible base policy 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.

We now describe the rollout algorithm. It starts at stage 0 and sequentially proceeds to the last stage. At stage k, it maintains a partial trajectory

$$T_k = (x_0, \overline{u}_0, \overline{x}_1, \ldots, \overline{u}_{k-1}, \overline{x}_k) \tag{4.5}$$

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

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

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 policy 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 $u_k \in \overline{U}_k(\overline{x}_k)$

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

[We assume that the existence of a minimizing $\overline{u}_k$ is guaranteed when $\overline{U}_k(\overline{x}_k)$ is nonempty; this is true for example when $U_k(\overline{x}_k)$, and hence also $\overline{U}_k(\overline{x}_k)$, is finite.] The algorithm then forms the partial trajectory T_{k+1} by adding $(\overline{u}_k, \overline{x}_{k+1})$ to T_k, where

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

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

$$T_k^c(u_k) = T_k \cup (u_k) \cup H(f_k(\overline{x}_k, u_k)), \tag{4.6}$$

$$\overline{U}_k(\overline{x}_k) = \{u_k \mid u_k \in U_k(\overline{x}_k), T_k^c(u_k) \in C\}. \tag{4.7}$$

The rollout algorithm selects at stage k

$$\overline{u}_k \in \arg \min_{u_k \in \overline{U}_k(\overline{x}_k)} [g_k(\overline{x}_k, u_k) + \tilde{J}(f_k(\overline{x}_k, u_k))], \tag{4.8}$$

and extends the current partial trajectory by setting

$$T_{k+1} = T_k \cup (\overline{u}_k, \overline{x}_{k+1}), \tag{4.9}$$

where

$$\overline{x}_{k+1} = f_k(\overline{x}_k, \overline{u}_k). \tag{4.10}$$

Figure 1 provides an illustration of the algorithm.

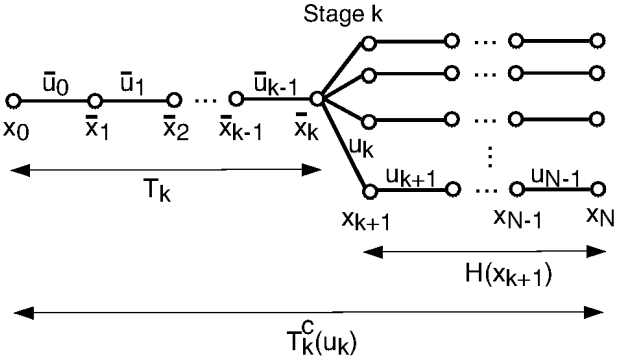

Fig. 1. Illustration of the rollout algorithm for constrained deterministic DP problems. At stage k, and given the current partial trajectory T_k that starts at x_0 and ends at $\overline{x}_k$, it considers all possible next states $x_{k+1} = f_k(\overline{x}_k, u_k), u_k \in U_k(\overline{x}_k)$, and runs the base policy 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 $(\overline{u}_k, \overline{x}_{k+1})$.

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 4.1. The base policy 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 policy 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 policy 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 policy starting from x_k need not be equal. However, they are equal when the base policy is sequentially consistent, that is

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

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

$$g_k(x_k, \mu_k(x_k)) + \tilde{J}(f_k(x_k, \mu_k(x_k))) = \tilde{J}(x_k), \ \forall x_k, k. \tag{4.11}$$

Note that base policies that are of the 'greedy' type tend to be sequentially consistent, as explained in Ref. [13], Section 6.4. The next definition provides an extension of the notion of sequential consistency.

Definition 4.2. The base policy 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)

$$g_k(x_k, u_k) + \tilde{J}(x_{k+1}) \leq \tilde{J}(x_k). \tag{4.12}$$

(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

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

Properties (1) and (2) of the preceding definition provide some basic guarantees for the use of the control u_k dictated by the base policy at state x_k. In particular, property (1) guarantees that when u_k is used, the cost $\tilde{J}(x_{k+1})$ of the base policy starting from the new state x_{k+1} will not be 'excessive', in the sense that it will be no more than the cost $\tilde{J}(x_k) - g_k(x_k, u_k)$ 'predicted' at state x_k. Property (2) guarantees that when u_k is used, the base policy will maintain feasibility starting from the new state x_{k+1}.

Note that if the base policy is sequentially consistent, it is also sequentially improving [cf. Eqs (4.11) and (4.12)]. An example of an interesting case where the sequential improvement property holds arises in MPC (see the next section). There, the base policy drives the system to a cost-free and absorbing state within a given number of stages, while minimizing the cost over these stages. It can be verified that for a system with a cost-free and absorbing state, a base policy of this type is sequentially improving. For another case where the base policy 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

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

[cf. Eq. (4.4)]. Let $\tilde{C}^m(x_k)$ be the value of the m-th 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 policy starting from x_k, that is

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

Then, it can be seen that the base policy is sequentially improving if in addition to Eq. (4.12), the partial trajectory $H(x_k)$ satisfies

$$g_k^m(x_k, u_k) + \tilde{C}^m(x_{k+1}) \leq \tilde{C}^m(x_k), \quad m = 1, \ldots, M. \tag{4.13}$$

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

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

and from Eq. (4.13), it follows that

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

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.

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

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

that are feasible [cf. Eqs (4.6) and (4.7)], and the trajectory $\overline{T}_k^c$ within this set that corresponds to the control $\overline{u}_k$ chosen by the rollout algorithm, that is

$$\overline{T}_k^c = T_k^c(\overline{u}_k) = T_k \cup (\overline{u}_k) \cup H(\overline{x}_{k+1}), \tag{4.14}$$

where

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

The result asserts that as k increases, the cost of the corresponding trajectories $\overline{T}_k^c$ cannot increase.

Proposition 4.1. Assume that the base policy is sequentially improving. Then for each k, the set $\overline{U}_k(\overline{x}_k)$ is nonempty, and

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

where the trajectories $\overline{T}_k^c$ are given by Eq. (4.14) for all k.

Proof. Let the trajectory generated by the base policy 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

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

where the inequality follows by the sequential improvement assumption [cf. Eq. (4.12)]. Since

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

it follows by combining the preceding two relations and the definition of $\overline{T}_0^c$ [cf. Eq. (4.14)] that

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

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

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

where the inequality follows by the sequential improvement assumption. Since

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

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. □

Proposition 4.1 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 4.2. If the base policy 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 policy starting from x_0.

By a simple extension of the argument used to show Proposition 4.2, we can also show that when the base policy is sequentially improving, then the trajectory $(x_0, \overline{u}_0, \overline{x}_1, \ldots, \overline{u}_{N-1}, \overline{x}_N)$ produced by the rollout algorithm satisfies for all $k = 1, \ldots, N-1$,

$$\tilde{J}(\overline{x}_k) \geq g_N(\overline{x}_N) + \sum_{i=k}^{N-1} g_i(\overline{x}_i, \overline{u}_i).$$

In words, the cost-to-go of the rollout algorithm starting from state $\overline{x}_k$ is no worse than the cost-to-go of the base policy starting from $\overline{x}_k$. Thus, $\tilde{J}(\overline{x}_k)$ provides a readily computable upper bound to the cost-to-go of the rollout algorithm starting from $\overline{x}_k$.

We will now discuss some variations and extensions of the rollout algorithm for constrained deterministic problems. Let us consider the case where the sequential improvement assumption is not satisfied. Then, even if the base policy 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. (4.7)] 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 Ref. [3] for the case of an unconstrained DP problem (see also [13], 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 policy 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

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

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 $u_k \in \tilde{U}_k(\overline{x}_k)$

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

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 policy 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 policy 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 proofs of Propositions 4.1 and 4.2, that even when the base policy 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 policy starting from x_0.

We finally consider the case where C is specified by the time-additive constraints

$$g_N^m(x_N) + \sum_{k=0}^{N-1} g_k^m(x_k, u_k) \le b^m, \quad m = 1, \ldots, M, \tag{4.15}$$

[cf. Eq. (4.4)], representing restrictions on M resources. Then, it is natural for the base policy 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

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

$$y_0^m = 0,$$

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

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

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

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

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

The time-additive constraints (4.15) are then reformulated as $y_N^m \le b^m$, $m = 1, \ldots, M$. The rollout algorithm described earlier, when used in conjunction with this reformulated problem, allows for base policies whose generated trajectories depend not only on x_k but also on $y_k^1, \ldots, y_k^M$, and brings to bear Propositions 4.1 and 4.2.

4.3. Open-Loop Feedback Control

We will now discuss a classical suboptimal control scheme, which turns out to be a special case of the rollout algorithm. We have so far focused on problems of perfect state information where the state x_k is observed exactly. In an imperfect state information problem, control at stage k is applied with knowledge of the information vector

$$I_k = (z_1, \ldots, z_k, u_0, \ldots, u_{k-1}),$$

where for all k, z_k is an observation obtained at stage k, and related to the current state x_k and the preceding control u_{k-1} through a given conditional probability distribution. Following the observation z_k, a control u_k is chosen by the controller with knowledge of the information I_k and the associated the conditional probability distribution $P_{x_k|I_k}$. Then, a cost $g(x_k, u_k)$ is incurred, where x_k is the current (hidden) state. While problems of imperfect state information are very hard to solve optimally, they can be reduced to problems of perfect state information whose state is the conditional distribution $P_{x_k|I_k}$ (see e.g., [13]).

Generally, in a problem with imperfect state information, the performance of the optimal policy improves when extra information is available. However, the use of this information may make the DP calculation of the optimal policy intractable. This motivates a suboptimal policy, based on a more tractable computation that in part ignores the availability of extra information. This policy was first suggested by Dreyfus [21], and is known as the open-loop feedback controller (OLFC for short). It uses the

current information vector I_k to determine $P_{x_k|I_k}$, but it calculates the control u_k as if no further measurements will be received, by using an open-loop optimization over the future evolution of the system. In particular, u_k is determined as follows:

(1) Given the information vector I_k, compute the conditional probability distribution $P_{x_k|I_k}$ (in the case of perfect state information, where I_k includes x_k, this step is unnecessary).

(2) Find a control sequence $\{\overline{u}_k, \overline{u}_{k+1}, \ldots, \overline{u}_{N-1}\}$ that solves the open-loop problem of minimizing

$$E\{g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, u_i, w_i) \mid I_k\}$$

subject to the constraints

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

(3) Apply the control input

$$\overline{\mu}_k(I_k) = \overline{u}_k.$$

Thus the OLFC uses at time k the new measurement z_k to calculate the conditional probability distribution $P_{x_k|I_k}$. However, it selects the control input as if future measurements will be disregarded.

In any suboptimal control scheme, one would like to be assured that measurements are advantageously used. By this we mean that the scheme performs at least as well as any open-loop policy that uses a sequence of controls that is independent of the values of the measurements received. An optimal open-loop policy can be obtained by finding a sequence $\{u_0^*, u_1^*, \ldots, u_{N-1}^*\}$ that minimizes

$$\overline{J}(u_0, u_1, \ldots, u_{N-1}) = E\{g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k)\}$$

subject to the constraints

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

A nice property of the OLFC is that it performs at least as well as an optimal open-loop policy, as shown by the following proposition. In contrast, the CEC does not have this property (for example, for a one-stage problem, the optimal open-loop controller and the OLFC are both optimal, but the CEC may be strictly suboptimal).

Proposition 4.3. The cost $J_{\overline{\pi}}$ corresponding to an OLFC satisfies

$$J_{\overline{\pi}} \leq J_0^*,$$

where J_0^* is the cost corresponding to an optimal open-loop policy.

The preceding proposition shows that the OLFC uses the measurements advantageously even though it selects at each period the present control input as if no further measurements will be taken in the future. A proof from first principles was given in Bertsekas [11] (this proof is reproduced in Ref. [13]; it was extended by White and Harrington [49]). However, it is much simpler to argue that the proposition is a special case of the cost improvement property of the rollout algorithm (cf. Example 3.1), based on the following observation.

Let us view the given problem of imperfect state information as a problem of perfect state information whose 'state' at the k-th stage is the distribution $P_{x_k|I_k}$. Let us also view the optimal open-loop policy starting at the next 'state' $P_{x_{k+1}|I_{k+1}}$ as a base policy. Then it can be seen that the OLFC is just the rollout algorithm corresponding to this base policy. According to the generic cost improvement property of rollout algorithms, the performance of the OLFC is no worse that the performance of the optimal open-loop policy (the base policy) starting from the initial state $P_{x_0|I_0}$. This is precisely the statement of Proposition 4.3.

We finally note that a form of suboptimal control that is intermediate between the optimal feedback controller and the OLFC is provided by a generalization of the OLFC called the *partial* OLFC (POLFC); see [13]. Similar to OLFC, this controller uses past measurements to compute $P_{x|I_k}$, but calculates the control input on the basis that *some* (but not necessarily all) of the measurements will in fact be taken in the future, and the remaining measurements will not be taken. This method often allows one to deal with those measurements that are troublesome and complicate the solution, while taking into account the future availability of other measurements that can be reasonably dealt with. The POLFC can also be viewed as a form of rollout policy whose base policy takes into account some (but not all) of the future measurements rather than being open-loop.

5. Model Predictive Control

MPC was motivated by the desire to introduce nonlinearities and constraints into the linear-quadratic control framework, while obtaining a suboptimal but stable closed-loop system. We will describe MPC for the more general case of a time-invariant nonlinear deterministic system and nonquadratic cost. The state and control belong to the Euclidean spaces $\Re^n$ and $\Re^m$, respectively, and may be constrained. In particular, the system is

$$x_{k+1} = f(x_k, u_k), \quad k = 0, 1, \ldots,$$

and the cost per stage is

$$g(x_k, u_k), \quad k = 0, 1, \ldots,$$

and is assumed nonnegative for all x_k and u_k. We impose state and control constraints

$$x_k \in X, \quad u_k \in U(x_k), \ k = 0, 1, \ldots,$$

and we assume that the set X contains the origin of $\Re^n$. Furthermore, if the system is at the origin, it can be kept there at no cost with control equal to 0, that is

$$0 \in U(0), \quad f(0, 0) = 0, \quad g(0, 0) = 0.$$

We want to derive a stationary feedback controller that applies control $\overline{\mu}(x)$ at state x, and is stable in the sense that $x_k \to 0$ and $\overline{\mu}(x_k) \to 0$ for all initial states $x_0 \in X$. Furthermore, we require that for all initial states $x_0 \in X$, the state of the closed-loop system

$$x_{k+1} = f(x_k, \overline{\mu}(x_k)),$$

satisfies the state and control constraints. Finally, to satisfy the stability requirement through a cost minimization, we require that the total cost of the feedback controller $\overline{\mu}$ over an infinite number of stages is finite:

$$\sum_{k=0}^{\infty} g(x_k, \overline{\mu}(x_k)) < \infty, \tag{5.1}$$

and that the nonnegative function g is such that the preceding relation implies that $x_k \to 0$ and $\overline{\mu}(x_k) \to 0$. A primary example is a quadratic cost per stage,

$$g(x, u) = x'Qx + u'Ru,$$

where the matrices Q and R are positive definite and symmetric.

In order for such a controller to exist, it is evidently sufficient [in view of the assumptions $f(0, 0) = 0$ and $g(0, 0) = 0$] that *there exists a positive integer m such that for every initial state $x_0 \in X$, one can find a sequence of controls $u_k, k = 0, 1, \ldots, m-1$, which drive to 0 the state x_m of the system at time m, while keeping all the preceding states $x_1, x_2, \ldots, x_{m-1}$ within X and satisfying the control constraints $u_0 \in U(x_0), \ldots, u_{m-1} \in U(x_{m-1})$*. We refer to this as the *constrained controllability assumption*. In practical applications, the constrained controllability assumption can often be checked easily. Alternatively, the state and control constraints can be constructed in a way that the assumption is satisfied using the methodology of reachability of target tubes; see the end of this section.

Let us now describe a form of MPC under the constrained controllability assumption. At each stage k and state $x_k \in X$, it solves an m-stage deterministic optimal control problem involving the same cost per stage and constraints, *and the requirement that the state after m stages be exactly equal to 0*. This is the problem of minimizing

$$\sum_{i=k}^{k+m-1} g(x_i, u_i), \tag{5.2}$$

subject to the system equation constraints

$$x_{i+1} = f(x_i, u_i), \quad i = k, k+1, \ldots, k+m-1, \tag{5.3}$$

the state and control constraints

$$x_i \in X, \quad u_i \in U(x_i), \ i = k, k+1, \ldots, k+m-1, \tag{5.4}$$

and the terminal state constraint

$$x_{k+m} = 0. \tag{5.5}$$

By the constrained controllability assumption, this problem has a feasible solution. Let

$$\{\overline{u}_k, \overline{u}_{k+1}, \ldots, \overline{u}_{k+m-1}\}$$

be a corresponding optimal control sequence. The MPC applies at stage k the first component of this sequence,

$$\overline{\mu}(x_k) = \overline{u}_k,$$

and discards the remaining components.

We now show that the MPC satisfies the stability condition (5.1). Let $x_0, u_0, x_1, u_1, \ldots$ be the state and control sequence generated by MPC:

$$u_k = \overline{u}(x_k), \quad x_{k+1} = f(x_k, \overline{u}(x_k)), \ k = 0, 1, \ldots.$$

Denote by $\hat{J}(x)$ the optimal cost of the m-stage problem solved by MPC when at a state $x \in X$; cf. Eqs (5.2)–(5.5). Let also $\tilde{J}(x) = \infty$ be the optimal cost starting at x of a corresponding $(m-1)$-stage problem, i.e. the optimal value of the cost

$$\sum_{k=0}^{m-2} g(x_{k, u_k}),$$

where $x_0 = x$, subject to the constraints

$$x_k \in X, \quad u_k \in U(x_k), \ k = 0, 1, \ldots, m-2,$$

and

$$x_{m-1} = 0.$$

[For states $x \in X$ for which this problem does not have a feasible solution, we write $\tilde{J}(x) = \infty$.] Having one less stage in our disposal to drive the state to 0 cannot decrease the optimal cost, so we have for all $x \in X$

$$\hat{J}(x) \leq \tilde{J}(x). \tag{5.6}$$

From the definitions of $\hat{J}$ and $\tilde{J}$, we have for all k,

$$\min_{u \in U(x)} [g(x_k, u) + \tilde{J}(f(x_k, u))] = g(x_k, u_k) + \tilde{J}(x_{k+1}) = \hat{J}(x_k), \tag{5.7}$$

so using Eq. (5.6) with $x = x_{k+1}$, we see that

$$g(x_k, u_k) + \hat{J}(x_{k+1}) \leq \hat{J}(x_k), \quad k = 0, 1, \ldots. \tag{5.8}$$

Adding this equation for all k in the range $[0, K]$, where $K = 0, 1, \ldots$, we obtain

$$\hat{J}(x_{K+1}) + \sum_{k=0}^{K} g(x_k, u_k) \leq \hat{J}(x_0).$$

Since $\hat{J}(x_{K+1}) \geq 0$ (in view of the nonnegativity of g), it follows that

$$\sum_{k=0}^{K} g(x_k, u_k) \leq \hat{J}(x_0), \quad K = 0, 1, \ldots, \tag{5.9}$$

and taking the limit as $K \to \infty$,

$$\sum_{k=0}^{\infty} g(x_k, u_k) \leq \hat{J}(x_0) \leq \infty.$$

This shows the stability condition (5.1).

We note that *the one-step lookahead function $\tilde{J}$ implicitly used by MPC [cf. Eq. (5.7)] is the cost corresponding to a certain base policy*. This policy is defined by the open-loop control sequence that drives to 0 the state after $m - 1$ stages and keeps the state at 0 thereafter, while observing the state and control constraints $x_k \in X$ and $u_k \in U(x_k)$, and minimizing the cost. Thus, we can also view MPC as a rollout algorithm with the base policy defined by the sequence just described. This base policy may not be sequentially consistent, but it is sequentially improving (cf. Definitions 4.1 and 4.2). The reason is that if $(x_{k+1}, u_{k+1}, x_{k+2}, u_{k+2}, \ldots, x_{k+m-1}, u_{k+m-1}, 0)$ is the sequence generated by the $(m-1)$-stage base policy starting from state x_{k+1}, we have

$$g(x_{k+1}, u_{k+1}) + \tilde{J}(x_{k+2}) \leq \tilde{J}(x_{k+1});$$

this follows by an argument similar to the argument we used to show Eq. (5.8) based on Eqs (5.6) and (5.7). Thus, property (1) of Definition 4.2 is satisfied, while the feasibility property (2) of that definition is also satisfied because of the structure of the constraints of the problem solved at each stage by MPC. In fact the stability property of MPC is a special case of the cost improvement property of rollout algorithms that employ a sequentially improving base policy (cf. Proposition 4.2), which under our assumptions for the cost per stage g, implies that *if the base policy results in a stable closed-loop system, the same is true for the corresponding rollout algorithm*. Note that the constrained controllability assumption is used to satisfy the feasibility assumption on the base policy within the rollout context of Section 4.

The connection between MPC and rollout is conceptually useful, and may lead to new algorithms for specialized or different types of control problems. For example, it can be exploited to derive MPC schemes involving more complex constraints in analogy with the rollout algorithm given in Section 4.

Looking back into the argument that we used to prove stability [cf. Eqs (5.6)–(5.9)], we see that to obtain the stability property (5.1), we do not need to solve the m-stage optimal control problem (5.2)–(5.5). It is sufficient instead to apply at any stage k and state $x_k \in X$, a control $u_k = \overline{u}_k$, where $\overline{u}_k$ is the first element of a sequence $\{\overline{u}_k, \overline{u}_{k+1}, \ldots, \overline{u}_{k+m-1}\}$ generated by an m-stage base policy starting at x_k (m may be dependent on x_k, but for simplicity we do not show this dependence). This control sequence, together with the corresponding generated state sequence $\{x_k, x_{k+1}, \ldots, x_{k+m}\}$, must have the following two properties:

(1) $x_{k+1} \in X$.
(2) $g(x_k, \overline{u}_k) + \hat{J}(x_{k+1}) \leq \hat{J}(x_k)$, where $\hat{J}(x)$ is some function of x that is nonnegative over the set X [for example, $\hat{J}(x)$ can be the cost incurred by the base policy over m stages starting from x, cf. Eq. (5.8)].

Property (2) above is a Lyapounov stability-type of condition, where $\hat{J}(x)$ is the Lyapounov function. Note that property (1) is weaker than the property $x_i \in X$ for all $i = k+1, \ldots, k+m-1$ satisfied by the sequence $\{\overline{u}_k, \overline{u}_{k+1}, \ldots, \overline{u}_{k+m-1}\}$ that is generated by MPC at stage k. However, the later property is also

needed to show property (2) [cf. Eq. (5.8)] via the argument of Eqs (5.6) and (5.7).

The MPC scheme that we have described is just the starting point for a broad methodology with many variations. For example, the m-stage problem solved by MPC at each stage may be modified so that instead of requiring that the state be 0 after m stages, one may use a large penalty for the state being nonzero after m stages. Then, the preceding analysis goes through, as long as the terminal penalty is chosen so that Eq. (5.6) is satisfied. In another variant, instead of aiming to drive the state to 0 after m stages, one aims to reach a sufficiently small neighborhood of the origin, within which a stabilizing controller, designed by other methods, may be used. This variant is also well-suited for taking into account disturbances described by set membership, as we now proceed to explain.

5.1. MPC with Set-Membership Disturbances

To extend the MPC methodology to the case where there are disturbances w_k in the system equation

$$x_{k+1} = f(x_k, u_k, w_k),$$

we must first modify the stability objective. The reason is that in the presence of disturbances, the stability condition (5.1) is impossible to meet. A reasonable alternative is to introduce a set-membership constraint $w_k \in W(x_k, u_k)$ for the disturbance and a target set T for the state, and to require that the controller specified by MPC drives the state to T with finite cost.

To formulate the MPC, we assume that $T \subset X$, and that once the system state enters T, we will use some control law $\tilde{\mu}$ that keeps the state within T for all possible values of the disturbances, that is

$$f(x, \tilde{\mu}(x), w) \in T, \quad \text{for all } x \in T, \quad w \in W(x, \tilde{\mu}(x)). \tag{5.10}$$

Finding such a target set T and control law $\tilde{\mu}$ can be addressed via the methodology of infinite time reachability, and will be discussed at the end of this section.

Once we specify the target set T, we can view it essentially as a cost-free and absorbing state, similar to our view of the origin in the earlier deterministic context. Consistent with this interpretation, we introduce the stage cost function

$$g(x,u) = \begin{cases} \overline{g}(x,u) & \text{if } x \notin T, \\ 0 & \text{if } x \in T, \end{cases}$$

where $\overline{g}$ is a nonnegative function with the property that for any sequence $\{x_0, u_0, x_1, u_1, \ldots\}$ generated by the system, the condition

$$\sum_{k=0}^{\infty} \overline{g}(x_k, u_k) < \infty,$$

implies that $x_k \to 0$ and $u_k \to 0$ [for example, $\overline{g}$ can be quadratic of the form $\overline{g}(x,u) = x'Qx + u'Ru$, where Q and R are positive definite, symmetric matrices].

The MPC is now defined as follows: at each stage k and state $x_k \in X$ with $x_k \notin T$, it solves the m-stage minimax control problem of finding a policy $\hat{\mu}_k, \hat{\mu}_{k+1}, \ldots, \hat{\mu}_{k+m-1}$ that minimizes

$$\max_{\substack{w_i \in W(x_i, \hat{\mu}(x_i)), \\ i=k,k+1,\ldots,k+m-1}} \sum_{i=k}^{k+m-1} g(x_i, \hat{\mu}(x_i)),$$

subject to the system equation constraints

$$x_{i+1} = f(x_i, \hat{\mu}(x_i), w_i), \quad i = k, k+1, \ldots, k+m-1,$$

the control and state constraints

$$x_i \in X, \quad \hat{\mu}(x_i), \in U(x_i), \; i = k, k+1, \ldots, k+m-1,$$

and the terminal state constraint

$$x_i \in T, \quad \text{for some } i \in [k+1, k+m].$$

These constraints must be satisfied for all disturbance sequences satisfying

$$w_i \in W(x_i, \hat{\mu}_i(x_i)), \quad i = k, k+1, \ldots, k+m-1.$$

The MPC applies at stage k the first component of the policy $\hat{\mu}_k, \hat{\mu}_{k+1}, \ldots, \hat{\mu}_{k+m-1}$ thus obtained,

$$\overline{\mu}(x_k) = \hat{\mu}_k(x_k),$$

and discards the remaining components. For states x within the target set T, the MPC applies the control $\tilde{\mu}(x)$ that keeps the state within T, as per Eq. (5.10) , at no further cost [$\overline{\mu}(x) = \tilde{\mu}(x)$ for $x \in T$].

We make a constrained controllability assumption, namely that the *m-stage minimax problem solved at each stage by MPC has a feasible solution for all* $x_k \in X$ with $x_k \notin T$ (this assumption can be checked using the target tube reachability methods, described at the end of this section). Note that this problem is a potentially difficult minimax control problem, which generally must be solved by DP (see e.g., Section 1.6 of Ref. [13]).

The stability analysis of MPC (in the modified sense of reaching the target set T with finite cost, for all possible disturbance values) is similar to the one given earlier in the absence of disturbances. Furthermore, we can view MPC in the presence of disturbances as a special case of a rollout algorithm, suitably modified to take account of the set-membership description of the disturbances. Let us provide the details of this analysis.

We first show that $\overline{\mu}$ attains reachability of the target tube $\{X, X, \ldots\}$ in the sense that

$$f(x, \overline{\mu}(x), w) \in X, \quad \text{for all } x \in X \text{ and } w \in W(x, \overline{\mu}(x)). \tag{5.11}$$

Indeed, for $x \in T$, MPC applies the control $\tilde{\mu}(x)$, which by assumption keeps the next state within T, and hence also within X (since $T \subset X$). Hence, Eq. (5.11) is satisfied for $x \in T$. For $x \in X$ with $x \notin T$, the constraints of the m-stage minimax problem solved by MPC starting from x include Eq. (5.11). Hence the reachability condition (5.11) is satisfied. (Note that the constrained controllability assumption that the m-stage minimax problem solved at each stage by MPC has a feasible solution is critical for this analysis.)

Consider now any sequence $\{x_0, u_0, x_1, u_1, \ldots\}$ generated by MPC [i.e. $x_0 \in X$, $x_0 \notin T$, $u_k = \overline{\mu}(x_k)$, $x_{k+1} = f(x_k, u_k, w_k)$ and $w_k \in W(x_k, u_k)$]. We will show that

$$\sum_{k=0}^{K_T - 1} g(x_k, u_k) \leq \hat{J}(x_0) < \infty, \tag{5.12}$$

where $\hat{J}(x)$ is the optimal cost starting at state $x \in X$ of the m-stage minimax control problem solved by MPC, and K_T is the smallest integer k such that $x_k \in T$ (with $K_T = \infty$ if $x_k \notin T$ for all k). Note that as a consequence of Eq. (5.12) there are two possibilities:

(1) The state reaches the target set T at some finite time K_T, in which case it is kept within T for all subsequent times, by the definition of MPC.
(2) The state never reaches the target set T ($K_T = \infty$), in which case we have $g(x_k, u_k) = \overline{g}(x_k, u_k)$ for all k, and the infinite horizon cost $\sum_{k=0}^{\infty} \overline{g}(x_k, u_k)$ is finite. By our assumption regarding $\overline{g}$, this implies that $x_k \to 0$ and $u_k \to 0$.

In either case (1) or (2), Eq. (5.12) can be viewed as a property of stability in the presence of disturbances (assuming that T is a bounded set).

To show Eq. (5.12), we argue similar to the case where there are no disturbances. Let us consider an optimal control problem that is similar to the one solved at each stage by MPC, but has one stage less. In particular, given $x \in X$ with $x \notin T$, consider the minimax control problem of finding a policy $\hat{\mu}_0, \hat{\mu}_1, \ldots, \hat{\mu}_{m-2}$ that minimizes

$$\max_{\substack{w_i \in W(x_i, \hat{\mu}(x_i)), \\ i=0,1,\ldots,m-2}} \sum_{i=0}^{m-2} g(x_i, \hat{\mu}_i(x_i)), \tag{5.13}$$

subject to the system equation constraints

$$x_{i+1} = f(x_i, \hat{\mu}_i(x_i), w_i), \quad i = 0, 1, \ldots, m-2, \tag{5.14}$$

the control and state constraints

$$x_i \in X, \quad \hat{\mu}_i(x_i) \in U(x_i), \quad i = 0, 1, \ldots, m-2, \tag{5.15}$$

and the terminal state constraint

$$x_i \in T, \quad \text{for some } i \in [1, m-1]. \tag{5.16}$$

These constraints must be satisfied for all disturbance sequences with

$$w_i \in W(x_i, \hat{\mu}_i(x_i)), \quad i = 0, 1, \ldots, m-2. \tag{5.17}$$

Let $\tilde{J}(x_0)$ be the corresponding optimal value, and define $\tilde{J}(x_0) = 0$ for $x_0 \in T$, and $\tilde{J}(x_0) = \infty$ for all $x_0 \notin T$ for which the problem has no feasible solution. It can be seen that the control $\overline{\mu}(x)$ applied by MPC at a state $x \in X$ with $x \notin T$, minimizes over $u \in U(x)$

$$\max_{w \in W(x,u)} [g(x, u) + \tilde{J}(f(x, u, w))].$$

By using the fact $\hat{J}(x) \leq \tilde{J}(x)$, it follows that for all $x \in X$ with $x \notin T$, we have

$$\max_{w \in W(x,u)} [g(x, \overline{\mu}(x)) + \hat{J}(f(x, \overline{\mu}(x), w))] \leq \hat{J}(x).$$

Hence, for all k such that $x_k \in X$ with $x_k \notin T$, we have

$$g(x_k, u_k) + \hat{J}(x_{k+1}) \leq \hat{J}(x_k),$$

where $\hat{J}(x_{k+1}) = 0$ if $x_{k+1} \in T$, and by adding over $k = 0, 1, \ldots, K_T - 1$, we obtain the desired result (5.12).

Note that similar to the case where there are no disturbances, we can interpret MPC as a rollout algorithm with a base policy defined as follows: for $x \in T$, the base policy applies $\tilde{\mu}(x)$, and for $x \in X$ with $x \notin T$, it applies the first element of a sequence that solves the $(m-1)$-stage problem described above, i.e. minimizes the cost (5.13) subject to the constraints (5.14)–(5.17).

5.2. MPC and Infinite-Time Reachability

We will now describe the connection of MPC, and the methodology of reachability of target sets and tubes, first introduced and developed for discrete-time systems by the author in his Ph.D. thesis [9], and subsequent papers [7,10]. Reachability, as well as the broader subject of estimation and control for systems with a set-membership description of the uncertainty, stayed outside mainstream control theory and practice for a long time, but have received renewed attention since the late 1980s, in the context of robust control and MPC. We refer to the surveys by Deller [20], Kosut et al. [27], Blanchini [15] and Mayne [38], which give many additional references. Section 4.6.2 of the textbook [13] provides an introduction to the subject, which is relevant to the material that follows.

Consider the system

$$x_{k+1} = f_k(x_k, u_k, w_k),$$

where w_k is known to belong to a given set $W_k(x_k, u_k)$, which may depend on the current state x_k and control u_k. A basic reachability problem is to find a policy $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$ with $\mu_k(x_k) \in U_k(x_k)$ for all x_k and k, such that for each $k = 1, 2, \ldots, N$, the state x_k of the closed-loop system

$$x_{k+1} = f_k(x_k, \mu_k(x_k), w_k)$$

belongs to a given set X_k, called the *target set at time k*.

We may view the set sequence $\{X_1, X_2, \ldots, X_N\}$ as a 'tube' within which the state must stay, even under the worst possible choice of the disturbances w_k from within the corresponding sets $W_k(x_k, \mu_k(x_k))$. If there exists a policy $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$ that keeps the state x_k within X_k for all $k = 1, \ldots, N-1$, starting from a given initial state x_0, we say that *the tube* $\{X_1, X_2, \ldots, X_N\}$ *is reachable from* x_0.

Turning back to the MPC formulation, we note that the constrained controllability assumption is in effect an assumption about reachability of a certain target tube. In particular, for a deterministic system, this assumption requires that the state constraint set X is such that for all initial states within X, there exists a control sequence that keeps the state of the system within x for the next $m-1$ stages, and drives the state to 0 at the m-th stage. This is equivalent to assuming that the tube $\{X_1, X_2, \ldots, X_m\}$ is reachable from all $x_0 \in X$, where

$$X_1 = X_2 = \cdots = X_{m-1} = X, \quad X_m = \{0\}.$$

For the case of a system driven by disturbances described by set membership, the constrained controllability assumption requires that we know two sets with favorable reachability properties: the target set T that the system aims at and the set X within which the state must stay as it approaches T. An equivalent statement of the assumption is that the tube $\{X_1, X_2, \ldots, X_m\}$ is reachable from all $x_0 \in X$, where

$$X_1 = X_2 = \cdots = X_{m-1} = X, \quad X_m = T,$$

and that the set T is *infinitely reachable*, in the sense that there exists a control law $\tilde{\mu}$ that keeps the state within T for all possible values of the disturbances, that is

$$f(x, \tilde{\mu}(x), w) \in T, \quad \text{for all } x \in T, w \in W(x, \tilde{\mu}(x)). \tag{5.18}$$

We may view the problem of infinite time reachability of a set T as a limit/infinite horizon version of the problem of reachability of a target tube: instead of a tube consisting of a finite number of sets, we consider a tube with an infinite number of sets of the form $\{T, T, \ldots\}$.

The computation as well as the definition of MPC critically depends on being able to solve the two reachability problems formulated above: reachability of a (finite horizon) target tube, and infinite time reachability of a set (or reachability of an infinite horizon target tube). One may formulate the problem of reachability of a target tube $\{X_1, X_2, \ldots, X_N\}$ as a minimax control problem, where the cost at stage k is

$$g_k(x_k) = \begin{cases} 0 & \text{if } x_k \in X_k, \\ 1 & \text{if } x_k \notin X_k. \end{cases}$$

With this choice, the optimal cost-to-go from a given initial state x_0 is the minimum number of violations of the target tube constraints $x_k \in X_k$ that can occur when the w_k are optimally chosen, subject to the constraint $w_k \in W_k(x_k, u_k)$, by an adversary wishing to maximize the number of violations. In particular, if $J_k(x_k) = 0$ for some $x_k \in X_k$, there exists a policy such that starting from x_k, the subsequent system states $x_i, i = k+1, \ldots, N$, are guaranteed to be within the corresponding sets X_i.

It can be seen that the set

$$\overline{X}_k = \{x_k | J_k(x_k) = 0\}$$

is the set that we *must* reach at time k in order to be able to maintain the state within the subsequent target sets. Accordingly, we refer to $\overline{X}_k$ as the *effective target set at time k*. We can generate the sets $\overline{X}_k$ with a

backwards recursion, first obtained in Ref. [9], which may be derived from the DP algorithm for minimax problems, but can also be easily justified from first principles. In particular, we start with

$$\overline{X}_N = X_N, \tag{5.19}$$

and for $k = 0, 1, \ldots, N-1$, we have

$$\overline{X}_k = \{x_k \in X_k \,|\, \text{there exists } u_k \in U_k(x_k) \text{ such that } f_k(x_k, u_k, w_k) \in \overline{X}_{k+1}, \text{ for all } w_k \in W_k(x_k, u_k)\}. \tag{5.20}$$

In general, it is not easy to characterize the effective target sets $\overline{X}_k$. However, a few special cases involving the linear system

$$x_{k+1} = A_k x_k + B_k u_k + w_k, \quad k = 0, 1, \ldots, N-1,$$

where A_k and B_k are given matrices, are amenable to exact or approximate computational solution. One such case is when the sets X_k are polyhedral, and the sets $U_k(x_k)$ and $W_k(x_k, u_k)$ are also polyhedral, and independent of x_k and u_k. Then the effective target sets are polyhedral and can be computed by linear programming methods.

Another case of interest is when the sets X_k are ellipsoids, and the sets $U_k(x_k)$ and $W_k(x_k, u_k)$ are also ellipsoids that do not depend on x_k and (x_k, u_k), respectively. In this case, the effective target sets $\overline{X}_k$ are not ellipsoids, but can be approximated by inner ellipsoids $\tilde{X}_k$ with

$$\tilde{X}_k \subset \overline{X}_k$$

(this requires that the ellipsoids U_k have sufficiently large size, for otherwise the target tube may not be reachable and the problem may not have a solution). Furthermore, the state trajectory $\{x_1, x_2, \ldots, x_N\}$ can be maintained within the ellipsoidal tube $\{\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_N\}$ by using a *linear control law* (see [13], p. 214). We refer to the author's Ph.D. thesis work [9] and the subsequent paper [7], for a more detailed analysis.

When the problem is stationary and f_k, X_k, U_k and W_k do not depend on k, as in the MPC formulation, an infinitely reachable set can often be obtained, at least in principle, by a form of value iteration, i.e. start with some (sufficiently large) target set Y, and sequentially construct an infinite sequence of corresponding effective target sets $\{\overline{Y}_k\}$ satisfying $\overline{Y}_0 = Y$ and

$$\overline{Y}_{k+1} = \{x \in \overline{Y}_k \,|\, \text{there exists } u \in U(x) \text{ with } f(x, u, w) \in \overline{Y}_k, \text{ for all } w \in W(x, u)\}.$$

Clearly, $\overline{Y}_k$ is the set of all initial states $x \in Y$ for which there exists a k-stage policy that keeps the state of the system within Y for the next k stages. We have

$$\overline{Y}_{k+1} \subset \overline{Y}_k, \quad \forall k,$$

and we may view the intersection

$$Y_\infty = \bigcap_{k=0}^{\infty} \overline{Y}_k$$

as the set within which the state can be kept for an arbitrarily large (but finite) number of time periods. Paradoxically, under some unusual circumstances, the set Y_∞ may not be infinitely reachable, i.e. there may exist states in Y_∞ starting from which it may be impossible to remain within Y_∞ for an indefinitely long horizon, i.e. an infinite number of time periods. Conditions that guarantee infinite reachability of Y_∞ have been investigated by the author in Ref. [10], and include compactness of the sets $Y, U(x)$ and $W(x, u)$, and continuity of the function f. Under these conditions, the set Y_∞ is the largest infinitely reachable set contained within Y. Refs [9,10] also focus on the case where the system is linear, and the sets Y, U and W are ellipsoids. For this case, these references provide a methodology, based on a Riccati-like equation for constructing infinitely reachable ellipsoidal inner approximations to Y_∞ and an associated linear control law that attains infinite time reachability.

6. Restricted Structure Policies

We will now introduce a general unifying suboptimal control scheme that contains as special cases several of the control schemes we have discussed: rollout, OLFC and MPC. The idea is to simplify the problem by selectively restricting the information and/or the controls available to the controller, thereby obtaining a restricted but more tractable problem structure, which can be used conveniently in a one-step lookahead context.

An example of such a structure is one where fewer observations are obtained, or one where the control constraint set is restricted to a single or a small number of given controls at each state. Generally, a restricted structure is associated with a problem where the optimal cost achievable is less favorable than in the given problem; this will be made specific in what follows. At each stage, we compute a policy that solves an optimal control problem involving the remaining stages and the restricted problem structure. The control applied at the given stage is the first component of the restricted policy thus obtained.

An example of a suboptimal control approach that uses a restricted structure is the OLFC, where one uses the information available at a given stage as the starting point for an open-loop computation (where future observations are ignored). Another example is the rollout algorithm, where at a given stage one restricts the controls available at future stages to be those applied by some suboptimal policy. Still another example is MPC, which under some conditions may be viewed as a form of rollout algorithm, as discussed in the preceding section.

For a problem with N stages, implementation of the suboptimal scheme to be discussed requires the solution of a problem involving the restricted structure at each stage. The horizon of this problem starts at the current stage, call it k, and extends up to the final stage N. This solution yields a control u_k for stage k and a policy for the remaining stages $k+1,\ldots,N-1$ (which must obey the constraints of the restricted structure). The control u_k is used at the current stage, while the policy for the remaining stages $k+1,\ldots,N-1$ is discarded. The process is repeated at the next stage $k+1$, using the additional information obtained between stages k and $k+1$.

Similarly, for an infinite horizon model, implementation of the suboptimal scheme requires, at each stage k, the solution of a problem involving the restricted structure and a (rolling) horizon of fixed length. The solution yields a control u_k for stage k and a policy for each of the remaining stages. The control u_k is then used at stage k, and the policy for the remaining stages is discarded. For simplicity in what follows, we will focus attention to the finite horizon case, but the analysis applies, with minor modifications, to infinite horizon cases as well.

Our main result is that the performance of the suboptimal control scheme is no worse than the one of the restricted problem, i.e. the problem corresponding to the restricted structure. This result unifies and generalizes our analysis for the OLFC (which is known to improve the cost of the optimal open-loop policy, cf. Section 4) for the rollout algorithm (which is known to improve the cost of the corresponding suboptimal policy, cf. Section 4) and for MPC (where under some reasonable assumptions, stability of the suboptimal closed-loop control scheme is guaranteed, cf. Section 5).

For simplicity, we focus on the imperfect state information framework for stationary finite-state Markov chains with N stages; the ideas apply to much more general problems with perfect and imperfect state information, as well problems with an infinite horizon. We assume that the system state is one of a finite number of states denoted $1,2,\ldots,n$. When a control u is applied, the system moves from state i to state j with probability $p_{ij}(u)$. The control u is chosen from a finite set U. Following a state transition, an observation is made by the controller. There is a finite number of possible observation outcomes, and the probability of each depends on the current state and the preceding control. The information available to the controller at stage k is the information vector

$$I_k = (z_1,\ldots,z_k,u_0,\ldots,u_{k-1}),$$

where for all i, z_i and u_i are the observation and control at stage i, respectively. Following the observation z_k, a control u_k is chosen by the controller and a cost $g(x_k,u_k)$ is incurred, where x_k is the current (hidden) state. The terminal cost for being at state x at the end of the N stages is denoted $G(x)$. We wish to minimize the expected value of the sum of costs incurred over the N stages.

We can reformulate the problem into a problem of perfect state information where the objective is to control the column vector of conditional probabilities

$$p_k = (p_k^1,\ldots,p_k^n)',$$

with

$$p_k^j = P(x_k = j \mid I_k), \quad j = 1,\ldots,n.$$

We refer to p_k as the *belief state*, and we note that it evolves according to an equation of the form

$$p_{k+1} = \Phi(p_k,u_k,z_{k+1}).$$

The function Φ represents an estimator, as discussed. The initial belief state p_0 is given.

The corresponding DP algorithm has the form

$$J_k(p_k) = \min_{u_k\in U}[p_k'g(u_k) + E_{z_{k+1}}\{J_{k+1}(\Phi(p_k,u_k,z_{k+1}))\,|\,p_k,u_k\}],$$

where $g(u_k)$ is the column vector with components $g(1,u_k),\ldots,g(n,u_k)$, and $p_k'g(u_k)$, the expected stage cost, is the inner product of the vectors p_k and $g(u_k)$. The algorithm starts at stage N, with

$$J_N(p_N) = p_N'G,$$

where G is the column vector with components $G(1),\ldots,G(n)$, and proceeds backwards.

We will also consider another control structure, where the information vector is

$$\bar{I}_k = (\bar{z}_1,\ldots,\bar{z}_k,u_0,\ldots,u_{k-1}), \quad k = 0,\ldots,N-1,$$

with $\overline{z}_i$ being some observation for each i (possibly different from z_i), and the control constraint set at each p_k is a given set $\overline{U}(p_k)$. The probability distribution of $\overline{z}_k$ given x_k and u_{k-1} is known, and may be different than the one of z_k. Also $\overline{U}(p_k)$ may be different from U [in what follows, we will assume that $\overline{U}(p_k)$ is a subset of U].

We introduce a suboptimal policy, which at stage k, and starting with the current belief state p_k, applies a control $\overline{\mu}_k(p_k) \in U$, based on the assumption that the future observations and control constraints will be according to the restricted structure. More specifically, this policy chooses the control at the typical stage k and state x_k as follows:

Restricted structure policy: At stage k and state x_k, apply the control

$$\overline{\mu}_k(p_k) = u_k,$$

where

$$(u_k, \hat{\mu}_{k+1}(\overline{z}_{k+1}, u_k), \ldots, \hat{\mu}_{N-1}(\overline{z}_{k+1}, \ldots, \overline{z}_{N-1}, u_k, \ldots, u_{N-2}))$$

is a policy that attains the optimal cost achievable from stage k onward with knowledge of p_k and with access to the future observations $\overline{z}_{k+1}, \ldots, \overline{z}_{N-1}$ (in addition to the future controls), and subject to the constraints

$$u_k \in U, \quad \mu_{k+1}(p_{k+1}) \in \overline{U}(p_{k+1}), \ldots, \mu_{N-1}(p_{N-1}) \in \overline{U}(p_{N-1}).$$

Let $\overline{J}_k(p_k)$ be the cost-to-go, starting at belief state p_k at stage k, of the restricted structure policy $\{\overline{\mu}_0, \ldots, \overline{\mu}_{N-1}\}$ just described. This is given by the DP algorithm

$$\overline{J}_k(p_k) = p_k' g(\overline{\mu}_k(p_k)) + E_{z_{k+1}}\{\overline{J}_{k+1}(\Phi(p_k, \overline{\mu}_k(p_k), z_{k+1})) \,|\, p_k, \overline{\mu}_k(p_k)\} \tag{6.1}$$

for all p_k and k, with the terminal condition $\overline{J}_N(p_N) = p_N' G$ for all p_N.

Let us also denote by $J_k^r(p_k)$ the optimal cost-to-go of the restricted problem, i.e. the one where the observations and control constraints of the restricted structure are used exclusively. This is the optimal cost achievable, starting at belief state p_k at stage k, using the observations $\overline{z}_i, i = k+1, \ldots, N-1$, and subject to the constraints

$$u_k \in \overline{U}(p_k), \ \mu_{k+1}(p_{k+1}) \in \overline{U}(p_{k+1}), \ldots, \mu_{N-1}(p_{N-1}) \in \overline{U}(p_{N-1}).$$

We will show, under certain assumptions to be introduced shortly, that

$$\overline{J}_k(p_k) \leq J_k^r(p_k), \quad \forall p_k, \ k = 0, \ldots, N-1,$$

and we will also obtain a readily computable upper bound to $\overline{J}_k(p_k)$. To this end, for a given belief vector p_k and control $u_k \in U$, we consider three optimal costs-to-go corresponding to three different patterns of availability of information and control restriction over the remaining stages $k+1, \ldots, N-1$. We denote:

$Q_k(p_k, u_k)$: The cost achievable from stage k onward starting with p_k, applying u_k at stage k, and optimally choosing each future control u_i, $i = k+1, \ldots, N-1$, with knowledge of p_k, the observations $z_{k+1}, \ldots, z_i$ and the controls $u_k, \ldots, u_{i-1}$, and subject to the constraint $u_i \in U$.

$Q_k^c(p_k, u_k)$: The cost achievable from stage k onward starting with p_k, applying u_k at stage k, and optimally choosing each future control $u_i, i = k+1, \ldots, N-1$, with knowledge of p_k, the observations $\overline{z}_{k+1}, \ldots, \overline{z}_i$, and the controls $u_k, \ldots, u_{i-1}$, and subject to the constraint $u_i \in \overline{U}(p_i)$. Note that this definition is equivalent to

$$Q_k^c(p_k, \overline{\mu}_k(p_k)) = \min_{u_k \in U} Q_k^c(p_k, u_k), \tag{6.2}$$

where $\overline{\mu}_k(p_k)$ is the control applied by the restricted structure policy just described.

$\hat{Q}_k^c(p_k, u_k)$: The cost achievable from stage k onward starting with p_k, applying u_k at stage k, optimally choosing the control u_{k+1} with knowledge of p_k, the observation z_{k+1}, and the control u_k, subject to the constraint $u_{k+1} \in U$, and optimally choosing each of the remaining controls $u_i, i = k+2, \ldots, N-1$, with knowledge of p_k, the observations $z_{k+1}, \overline{z}_{k+2}, \ldots, \overline{z}_i$, and the controls $u_k, \ldots, u_{i-1}$, and subject to the constraints $u_i \in \overline{U}(p_i)$.

Thus, the difference between $Q_k^c(p_k, u_k)$ and $Q_k(p_k, u_k)$ is due to the difference in the control constraint and the information available to the controller at *all* future stages $k+1, \ldots, N-1$ [$\overline{U}(p_{k+1}), \ldots, \overline{U}(p_{N-1})$ versus U, and $\overline{z}_{k+1}, \ldots, \overline{z}_{N-1}$ versus $z_{k+1}, \ldots, z_{N-1}$, respectively]. The difference between $Q_k^c(p_k, u_k)$ and $\hat{Q}_k^c(p_k, u_k)$ is due to the difference in the control constraint and the information available to the controller at the *single* stage $k+1$ [$\overline{U}(p_{k+1})$ versus U, and $\overline{z}_{k+1}$ versus z_{k+1}, respectively]. Our key assumptions are that

$$\overline{U}(p_k) \subset U, \quad \forall p_k, \ k = 0, \ldots, N-1, \tag{6.3}$$

$$Q_k(p_k, u_k) \leq \hat{Q}_k^c(p_k, u_k) \leq Q_k^c(p_k, u_k), \quad \forall p_k, \ u_k \in U, \ k = 0, \ldots, N-1. \tag{6.4}$$

Roughly, this means that the control constraint $\overline{U}(p_k)$ is more stringent than U, and the observations $\overline{z}_{k+1},\ldots,\overline{z}_{N-1}$ are 'weaker' (no more valuable in terms of improving the cost) than the observations $z_{k+1},\ldots,z_{N-1}$. Consequently, if Eqs (6.3) and (6.4) hold, we may interpret a controller that uses in part the observations $\overline{z}_k$ and the control constraints $\overline{U}(p_k)$, in place of z_k and U, respectively, as 'handicapped' or 'restricted'.

Let us denote:

$J_k(p_k)$: The optimal cost-to-go of the original problem, starting at belief state p_k at stage k. This is given by

$$J_k(p_k) = \min_{u_k\in U} Q_k(p_k,u_k). \tag{6.5}$$

$J_k^c(p_k)$: The optimal cost achievable, starting at belief state p_k at stage k, using the observations $\overline{z}_i, i=k+1,\ldots,N-1$, and subject to the constraints

$$u_k\in U,\quad \mu_{k+1}(p_{k+1})\in\overline{U}(p_{k+1}),\ldots,$$
$$\mu_{N-1}(p_{N-1})\in\overline{U}(p_{N-1}).$$

This is given by

$$J_k^c(p_k) = \min_{u_k\in U} Q_k^c(p_k,u_k), \tag{6.6}$$

and it is the cost that is computed when solving the optimization problem of stage k in the restricted structure policy scheme. Note that we have for all p_k,

$$\begin{aligned} J_k^r(p_k) &= \min_{u_k\in\overline{U}(p_k)} Q_k^c(p_k,u_k)\\ &\geq \min_{u_k\in U} Q_k^c(p_k,u_k) = J_k^c(p_k), \end{aligned} \tag{6.7}$$

where the inequality holds in view of the assumption $\overline{U}(p_k)\subset U$.

Our main result is as follows.

Proposition 6.1. Under the assumptions (6.3) and (6.4), there holds

$$J_k(p_k)\leq\overline{J}_k(p_k)\leq J_k^c(p_k)\leq J_k^r(p_k),$$
$$\forall p_k,\quad k=0,\ldots,N-1.$$

Proof. The inequality $J_k(p_k)\leq\overline{J}_k(p_k)$ is evident, since $J_k(p_k)$ is the optimal cost-to-go over a class of policies that includes the restricted structure policy $\{\overline{\mu}_0,\ldots,\overline{\mu}_{N-1}\}$. Also the inequality $J_k^c(p_k)\leq J_k^r(p_k)$ follows from the definitions [see Eq. (6.7)]. We prove the remaining inequality $\overline{J}_k(p_k)\leq J_k^c(p_k)$ by induction on k.

We have $\overline{J}_N(p_N)=J_N^c(p_N)=0$ for all p_N. Assume that for all p_{k+1}, we have

$$\overline{J}_{k+1}(p_{k+1})\leq J_{k+1}^c(p_{k+1}).$$

Then, for all p_k,

$$\begin{aligned} \overline{J}_k(p_k) &= p_k'g(\overline{\mu}_k(p_k))\\ &\quad +E_{z_{k+1}}\{\overline{J}_{k+1}(\Phi(p_k,\overline{\mu}_k(p_k),z_{k+1}))\,|\,p_k,\overline{\mu}_k(p_k)\}\\ &\leq p_k'g(\overline{\mu}_k(p_k))\\ &\quad +E_{z_{k+1}}\{J_{k+1}^c(\Phi(p_k,\overline{\mu}_k(p_k),z_{k+1}))\,|\,p_k,\overline{\mu}_k(p_k)\}\\ &= p_k'g(\overline{\mu}_k(p_k))\\ &\quad +E_{z_{k+1}}\{\min_{u_{k+1}\in U} Q_{k+1}^c(\Phi(p_k,\overline{\mu}_k(p_k),z_{k+1}),u_{k+1})\\ &\qquad |\,p_k,\overline{\mu}_k(p_k)\}\\ &= \hat{Q}_k^c(p_k,\overline{\mu}_k(p_k))\\ &\leq Q_k^c(p_k,\overline{\mu}_k(p_k))\\ &= J_k^c(p_k), \end{aligned}$$

where the first equality holds by Eq. (6.1), the first inequality holds by the induction hypothesis, the second equality holds by Eq. (6.6), the third equality holds by the definition of $\hat{Q}_k^c$, the second inequality holds by the assumption (6.4) and the last equality holds from the definition (6.2) of the restricted structure policy. The induction is complete. □

The main conclusion from the proposition is that the performance of the restricted structure policy $\{\overline{\mu}_0,\ldots,\overline{\mu}_{N-1}\}$ is no worse than the performance associated with the restricted control structure. Furthermore, at each stage k, the value $J_k^c(p_k)$, which is obtained as a byproduct of the on-line computation of the control $\overline{\mu}_k(p_k)$, is an upper bound to the cost-to-go $\overline{J}_k(p_k)$ of the suboptimal policy. This is consistent with our earlier results that show the cost improvement property of the rollout algorithm and the OLFC, and the stability property of MPC.

7. Conclusions

This paper has aimed to outline the connections between some suboptimal control schemes that have been the focus of much recent research. The underlying idea of these schemes is to start with a suboptimal/heuristic policy, and to improve it by using its cost-to-go (or a lower bound thereof) as an approximation to the optimal cost-to-go, the principal theme of the policy iteration method. While this viewpoint is natural in DP-based optimization, it is somewhat indirect within the control-oriented context of

MPC, where a principal issue is the stability of the closed-loop system. We have tried to emphasize the relation between the stability property of MPC and the cost improvement property of the underlying policy iteration methodology.

The connections between the various schemes described here may be helpful in better understanding their underlying mechanisms, and in extending the scope of their practical applications. In particular, the success of MPC in control engineering applications should motivate the broader use of rollout algorithms in practice, while the rollout and restricted policy cost improvement analyses should motivate the use of MPC methods in other problem domains, involving for example more complex state and control constraints. Finally, it should be mentioned that while in rollout algorithms for constrained DP, the issue of constructing a feasible base policy is left unresolved, in MPC it is addressed via the well-understood methodology of reachability of target tubes.

References

1. Abramson B. Expected-outcome: a general model of static evaluation. IEEE Trans Pattern Anal Machine Intell 1990; 12: 182–193
2. Barto A, Powell W, Si J (eds). Learning and approximate dynamic programming. IEEE Press, NY, 2004
3. Bertsekas DP, Tsitsiklis JN, Wu C. Rollout algorithms for combinatorial optimization. Heuristics 1997; 3: 245–262
4. Bertsekas DP, Castanon DA. Rollout algorithms for stochastic scheduling problems. Heuristics 1999; 5: 89–108
5. Bertsimas D, Demir R. An approximate dynamic programming approach to multi-dimensional knapsack problems. Manage Sci 2002; 4: 550–565
6. Bertsimas D, Popescu I. Revenue management in a dynamic network environment. Transportation Sci 2003; 37: 257–277
7. Bertsekas DP, Rhodes IB. On the minimax reachability of target sets and target tubes. Automatica 1971; 7: 233–247
8. Bertsekas DP, Tsitsiklis JN. Neuro-dynamic programming. Athena Scientific, Belmont, MA, 1996
9. Bertsekas DP. Control of uncertain systems with a set-membership description of the uncertainty, Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1971 (Available in scanned form from the author's www site)
10. Bertsekas DP. Infinite time reachability of state space regions by using feedback control. IEEE Trans Autom Control 1972; AC-17: 604–613
11. Bertsekas DP. On the solution of some minimax control problems. In: Proceedings of the 1972 IEEE decision and control conference, New Orleans, LA, 1972
12. Bertsekas DP. Differential training of rollout policies. In: Proceedings of the 35th Allerton conference on communication, control, and computing, Allerton Park, II, 1997, pp. 913–922
13. Bertsekas DP. Dynamic programming and optimal control. 3rd edn. Athena Scientific, Belmont, MA, 2005
14. Bertsekas DP. Rollout algorithms for constrained dynamic programming. Laboratory for Information and Decision Systems Report 2646, MIT, 2005
15. Blanchini F. Set invariance in control—a survey. Automatica 1999; 35: 1747–1768
16. Christodouleas,J.D. Solution Methods for Multiprocessor Network Scheduling Problems with Application to Railroad Operations, Ph.D. Thesis, Operations Research Center, Massachusetts Institute of Technology, 1997.
17. Chang HS, Givan RL, Chong EKP. Parallel rollout for online solution of partially observable Markov decision processes. Discrete Event Dynam Syst 2004; 14: 309–341
18. Camacho EF, Bordons C. Model predictive control. 2nd edn. Springer-Verlag, New York, NY, 2004
19. de Farias DP. Van Roy B. The linear programming approach to approximate dynamic programming. Operations Res 2003; 51: 850–865
20. de Farias DP, Van Roy B. On constraint sampling in the linear programming approach to approximate dynamic programming. Math Operations Res 2004; 29: 462–478
21. Deller JR. Set membership identification in digital signal processing. IEEE ASSP Mag 1989; October: 4–20
22. Dreyfus SD. Dynamic programming and the calculus of variations. Academic Press, NY, 1965
23. Findeisen R, Imsland L, Allgower F, Foss BA. State and output feedback nonlinear model predictive control: an overview. Eur J Control 2003; 9: 190–205
24. Ferris MC, Voelker MM. Neuro-dynamic programming for radiation treatment planning. Numerical Analysis Group Research Report NA-02/06, Oxford University Computing Laboratory, Oxford University, 2002
25. Ferris MC, Voelker MM. Fractionation in radiation treatment planning. Math Program B 2004; 102: 387–413
26. Guerriero F, Musmanno R. Label correcting methods to solve multicriteria shortest path problems. J Optim Theory Appl 2001; 111: 589–613
27. Guerriero F, Mancini M. A Co-operative Parallel Rollout Algorithm for the Sequential Ordering Problem, Parallel Computing, 2003; 29: 663–677
28. Jaffe JM. Algorithms for finding paths with multiple constraints. Networks 1984; 14: 95–116
29. Kosut RL, Lau MK, Boyd SP. Set-membership identification of systems with parametric and nonparametric uncertainty. IEEE Trans Autom Control 1992; AC-37: 929–941
30. Keerthi SS, Gilbert EG. Optimal, infinite horizon feedback laws for a general class of constrained discete time systems: stability and moving-horizon approximations. J Optim Theory Appl 1988; 57: 265–293
31. Konda VR, Tsitsiklis JN. Actor-critic algorithms. SIAM J Control Optim 2003; 42: 1143–1166
32. Konda VR. Actor-critic algorithms. Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, 2002

33. McGovern A, Moss E, Barto A. Building a basic building block scheduler using reinforcement learning and rollouts. Mach Learning 2002; 49: 141–160
34. Meloni C, Pacciarelli D, Pranzo M. A rollout metaheuristic for job shop scheduling problems. Ann Operations Res 2004; 131: 215–235
35. Mayne DQ, Rawlings JB, Rao CV, Scokaert POM. Constrained model predictive control: stability and optimality. Automatica 2000; 36: 789–814
36. Martin-Sanchez JM, Rodellar J. Adaptive predictive control. From the concepts to plant optimization. Prentice-Hall International (UK), Hemel Hempstead, Hertfordshire, UK, 1996
37. Maciejowski JM. Predictive control with constraints. Addison-Wesley, Reading, MA, 2002
38. Marbach P, Tsitsiklis JN. Simulation-based optimization of Markov reward processes. IEEE Trans Autom Control 2001; AC-46: 191–209
39. Martins EQV. On a multicriteria shortest path problem. Eur J Operational Res 1984; 16: 236–245
40. Mayne DQ. Control of constrained dynamic systems. Eur J Control 2001; 7: 87–99
41. Morari M, Lee JH. Model predictive control: Past, present, and future. Comput Chem Eng 1999; 23: 667–682
42. Qin SJ, Badgwell TA. A survey of industrial model predictive control Technology. Control Eng Practice 2003; 11: 733–764
43. Rantzer A. On relaxed dynamic programming in switching systems. IEE Proc Special Issue Hybrid Syst 2005 (to appear)
44. Rawlings JB. Tutorial overview of model predictive control. Control Syst Mag 2000; 20: 38–52
45. Secomandi N. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. Comput Operations Res 2000; 27: 1201–1225
46. Sutton R and Barto AG. Reinforcement Learning, MIT Press, Cambridge, MA, 1998
47. Secomandi N. A rollout policy for the vehicle routing problem with stochastic demands. Operations Res 2001; 49: 796–802
48. Secomandi N. Analysis of a rollout approach to sequencing problems with stochastic routing applications. J Heuristics 2003; 9: 321–352
49. Stewart BS, White CC. Multiobjective A^*. J ACM 1991; 38: 775–814
50. Tesauro G and Galperin GR. 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).
51. Tu F, Pattipati KR. Rollout strategies for sequential fault diagnosis. IEEE Trans Syst Man Cybernet A 2003; 86–99
52. Wu G, Chong EKP, Givan RL. Congestion control using policy rollout. In: Proceedings of the 2nd IEEE CDC, Maui, Hawaii. 2003; pp 4825–4830
53. White CC, Harrington DP. Application of Jensen's inequality to adaptive suboptimal design. J Optim Theory Appl 1980; 32: 89–99
54. Witsenhausen HS. Inequalities for the performance of suboptimal uncertain systems. Automatica 1969; 5: 507–512
55. Witsenhausen HS. On performance bounds for uncertain systems. SIAM J Control 1970; 8: 55–89
56. Yan X, Diaconis P, Rusmevichientong P, Van Roy B. Solitaire: man versus machine. Adv Neural Inform Process Syst 2005; 17: (to appear)