It is well-known that the quality of play of chess programs crucially depends on the size of the lookahead. Multistage lookahead has also been found to be useful to backgammon players, as will be discussed in Section 8.6. This indicates that in many types of problems, multistage lookahead should be much more effective than single stage lookahead. This improvement in performance must of course be weighed against the considerable increase in computation to obtain the decisions $\mu(i)$.

6.1.3 Rollout Policies

Suppose that through some method we have obtained a policy μ that we can simulate in our system. Then it is possible (at least in principle) to implement in real-time, instead of μ, the improved policy $\overline{\mu}$ that is obtained by a single *exact* policy improvement step from μ; that is, for all states i to be encountered in real-time operation of the system, we let

$$\overline{\mu}(i) = \min_{u \in U(i)} Q^{\mu}(i,u),$$

where

$$Q^{\mu}(i,u) = \sum_j p_{ij}(u)\big(g(i,u,j) + J^{\mu}(j)\big). \tag{6.5}$$

The key point here is that while our approximate DP method may provide only approximations to $Q^{\mu}(i,u)$, still the exact values of $Q^{\mu}(i,u)$ that are needed in the above policy improvement step can be calculated by Monte-Carlo simulation, as follows. Given any state i that is encountered in real-time operation of the system, and for every possible control $u \in U(i)$, we simulate trajectories that start at i, use decision u at the first stage, and use policy μ for all subsequent stages. The expected cost accumulated during such trajectories is the Q-factor $Q^{\mu}(i,u)$, and by simulating a large number of trajectories, an accurate estimate of $Q^{\mu}(i,u)$ is obtained. We then implement on the actual system a decision u with the smallest Q-factor. We refer to the policy $\overline{\mu}$ as the *rollout policy based on* μ.

Note that it is also possible to define rollout policies based on a policy μ, that make use of multistage (say, m-stage) lookahead. What is different here from the discussion in Section 6.1.2, is that instead of associating a cost-to-go $\tilde{J}(j,r)$ to every state j that can be reached in m steps, we use the exact cost-to-go $J^{\mu}(j)$, as computed by Monte Carlo simulation of several trajectories that start at j and follow policy μ. Clearly, such multistage lookahead involves much more on-line computation, but it may yield better performance than its single-stage counterpart. In what follows, we concentrate on rollout policies with single-stage lookahead.

The viability of a rollout policy depends on how much time is available to make the decision $\overline{\mu}(i)$ following the transition to state i and on how expensive the Monte Carlo evaluation of $Q^{\mu}(i,u)$ is. In particular, it must be possible to perform the Monte Carlo simulations and calculate the rollout control $\overline{\mu}(i)$ within the real-time constraints of the problem. If the problem is deterministic, a single simulation suffices, and the calculations are greatly simplified, but in general, the computational overhead can be substantial. However, it is worth emphasizing that the rollout policy $\overline{\mu}$ will always perform better than the current policy μ, as a consequence of the general results on the policy iteration method (Prop. 2.4).

It is possible to speed up the calculation of the rollout policy if we are willing to accept some potential performance degradation. Here are some possibilities:

(a) Use an approximation $\tilde{J}^\mu(\cdot, r)$ of J^μ to identify a few promising controls through a minimization of the form

$$\min_{u \in U(i)} \sum_j p_{ij}(u)\big(g(i,u,j) + \tilde{J}^\mu(j,r)\big),$$

calculate $Q^\mu(i,u)$ using fairly accurate Monte Carlo simulation for these controls, and approximate $Q^\mu(i,u)$ using relatively few simulation runs for the other controls. Adaptive variants of this approach are also possible, whereby we adjust the accuracy of the Monte Carlo simulation depending on the results of the computation.

(b) Use an approximate representation $\tilde{J}^\mu(\cdot, r)$ of J^μ to approximate $Q^\mu(i,u)$ by using simulation for N stages, and by estimating the cost of the remaining stages as $\tilde{J}^\mu(j_N, r)$, where j_N is the state after N stages.

(c) Identify a subset $\tilde{U}(i)$ of promising controls by using the procedures in (a) above or by using some heuristics, and use the control $\tilde{\mu}(i)$ given by

$$\tilde{\mu}(i) = \arg \min_{u \in \tilde{U}(i) \cup \{\mu(i)\}} Q^\mu(i,u), \tag{6.6}$$

where $Q^\mu(i,u)$ is obtained through accurate Monte-Carlo simulation.

Note that while the policies obtained from possibilities (a) and (b) may perform worse than μ, the policy $\tilde{\mu}$ in possibility (c) can be shown to perform at least as well as μ, i.e., $J^{\tilde{\mu}} \le J^\mu$. This is because Eq. (6.6) implies that $T_{\tilde{\mu}} J^\mu \le T_\mu J^\mu = J^\mu$, from which by the Monotonicity Lemma 2.1, we have $J^{\tilde{\mu}} = \lim_{t\to\infty} T_{\tilde{\mu}}^t J^\mu \le T_{\tilde{\mu}} J^\mu \le J^\mu$.

It is finally worth mentioning the broad generality of a rollout policy. It can be used to enhance the performance of *any* policy μ, no matter how obtained. In particular, μ does not need to be constructed using NDP techniques, and may be derived using heuristics.