Open-Loop Routing of N Arrivals to M Parallel Queues

Rodolfo A. Milito and Emmanuel Fernández-Gaucherand

Abstract—We consider the problem of routing N arrivals to M queues in parallel when no information other than the prior statistics are available for the decision process.

The work we present here differs from results in the literature in two significant ways: a) we address the nonstationary problem of open-loop routing for $N < \infty$ arrivals, and b) we do not decompose the problem into an allocation phase and a routing phase, but look instead for an overall optimal policy.

Exhaustive enumeration and dynamic programming turn out to be computationally feasible approaches for realistic values of the parameters N and M. Examination of the necessary conditions for optimality as given by Pontryagin’s maximum principle leads to the formulation of a policy iteration algorithm. Convergence of the algorithm in a finite number of iterations, as well as monotonicity results, are established. While only convergence to a local minimum is guaranteed, extensive computational experimentation points to its near-optimality. Some variants of the basic policy iteration algorithm are also discussed.

I. MOTIVATION

Routing to parallel queues is one of the most basic, nontrivial paradigms in normative queueing problems. The study of these paradigms is often pursued in the hope that they enhance our understanding of more complex, frequently intractable, queueing networks. As it turns out, the problem of routing to parallel queues is not only a nice paradigm to foster understanding, but it is a good model of many scenarios arising in computer and communication networks. Foremost among these problems is the distribution of calls in an intelligent network to multiple customer sites [1]-[4].

Most papers in the literature dealing with dynamic routing (see, e.g., the seminal paper of Hajek [5] or the excellent overview of Ephremides and Verdu [6]) assume that perfect state information is continuously available to the decision maker. This is a restrictive assumption, often violated in practice. Our ultimate goal is to give a rigorous treatment to the problem of routing arrivals to parallel queues when measurements are taken only every N arrivals. As a fundamental step in that direction, we address the problem of open-loop routing N arrivals to M queues with exponential servers.

So far, the literature regarding open-loop routing to parallel queues has considered the problem in steady state and has decomposed the problem into two phases:

a) Find the set $\{p_i, i = 1, 2, \ldots, M\}$ of routing probabilities to the different queues $\{Q_i, i = 1, 2, \ldots, M\}$ that would minimize the cost if randomized routing were used, and

b) Find the best routing sequence that in the long run assigns fraction $p_i$ of the arrivals to queue $Q_i$.

While intuitively and practically appealing, there is no guarantee that this decomposition yields optimal results. For problem b) in this decomposition framework, Hajek [7] found the most regular, optimal sequence for the case $M = 2$, and a lower bound on the cost for $M > 2$ [8]. Rosberg [9] proposed a golden ratio policy, for general $M \geq 2$, that achieves a cost close to Hajek’s lower bound. Arias and Levy [10] developed generalized round robin algorithms that are provably optimal for $M = 2$ and come very close to Hajek’s lower bound for $M > 2$. Section III of Levy et al. [2] proposes yet another extension of Hajek’s algorithm, to allow for fractions $\{p_i\}$ that change over time.

The work we present here differs from the existing approaches in two significant ways: a) we address the nonstationary problem of open-loop routing for $N < \infty$ arrivals, and b) we do not decompose the problem into an allocation phase and a routing phase, but instead look for an overall optimal policy.

II. PROBLEM FORMULATION

Consider the problem of routing N arrivals to M parallel queues ($Q_i, i = 1, 2, \ldots, M$), with the objective of minimizing a certain performance measure. The information available to the decision maker is limited to the system dynamics (service time and interarrival time distributions) and distribution of the initial number of customers in each queue. Under conditions on the interarrival and service times to be delineated in the sequel, the problem is cast as a completely unobservable controlled Markov chain. Therefore, an equivalent formulation can be given in terms of an information state (equivalently, sufficient statistics); see Bertsekas [11], Kumar and Varaiya [12], and Arapostathis et al. [13]. Let $\pi^{(n)}$ denote the marginal distribution of the number in queue i as seen by the centralized controller just before the nth arrival, given $\{a_0, a_1, \ldots, a_{n-1}\}$, where $a_n = i$ means that the n'th arrival is routed to $Q_i$. The finite action space is $A = \{1, 2, \ldots, M\}$. Given the finite horizon, the search for an optimal routing sequence can be restricted to the set of pure strategies without loss of generality. We concatenate the vectors $\pi^{(n)}$ together and call the resulting row vector $\pi^{(n)} = (\pi^{(0)}(n), \pi^{(1)}(n), \ldots, \pi^{(M)}(n))$. The evolution of the state of the M queues is given by

$$\pi_{n+1} = \pi_n P(a_n)$$  (2.1)

where the system matrix $P(a_n)$ is a block diagonal aggregate of the individual matrices $P^{(i)}(a_n)$. We emphasize that (2.1) is just a shorthand notation to describe the evolution of the M marginal distributions, rather than the joint probability distribution of the number in the M queues. The fact that the marginals evolve independently, and that they suffice for the problem at hand, relies on both the information pattern and the cost structure assumed in the paper.

Section II-B further elaborates on this crucial fact.

We assume that $Q_i, i \in \{1, 2, \ldots, M\}$ is served according to the first-in-first-out (FIFO) discipline by an exponential server that operates at rate $\mu^{(i)}$ and that the interarrival times are i.i.d., random variables of known distribution $F_a(x)$. Suppose that the underlying cost structure for queue $Q_i$ is given by an arbitrary function $\phi(j, a) \rightarrow c^{(i)}(j, a)$, where $j$ is the number in $Q_i$ and $a$ is the routing decision for the current arrival. Since $j$ is not known to the decision maker, an equivalent expected cost is formulated in terms of $\pi^{(i)}$ as the inner product

$$g^{(i)}(\pi^{(i)}(n), a_n) = \pi^{(i)}(n) c^{(i)}$$  (2.2)

$A$ could also include zero, namely, the decision to block an arrival, when such a decision is admissible.
where \( c_n^{(i)} = [c_n^{(i)}(0, a_n), c_n^{(i)}(1, a_n), \ldots] \). The cost function \( g(\pi_n, a_n) \) considered in our optimization problem is an additive function of the cost incurred by each individual queue, i.e.,

\[
g(\pi_n, a_n) = \sum_{i=1}^{M} g^{(i)}(\pi_n^{(i)}, a_n). \quad (2.3)
\]

Let us illustrate the cost structure with two performance measures of interest: the sojourn time of the incoming arrival, and the probability that in \( Q_i \) exceeds a given threshold \( t_i \). For the sojourn time cost, we have

\[
\epsilon_n^{(i)} = \begin{cases} 
[1, 2, 3, \cdots, t_i]^T / \mu^{(i)} & \text{if } a_n = i \\
[0, 0, 0, \cdots] & \text{otherwise.} 
\end{cases} \quad (2.4)
\]

If the interest is to penalize the probability that the number in \( Q_i \) exceeds threshold \( t_i \), one sets the first \( t_i \) components of the vector \( \epsilon_n^{(i)} \) to 0, and the rest to 1, if \( a_n \neq i \). If \( a_n = i \), the \( t_i \)th component is also set to zero.

Our goal is to minimize the sum of the costs seen by the first \( N \) arrivals given an initial distribution \( \pi_0 \) at the epoch of the first arrival, i.e.,

\[
G^{*}(\pi_0) := \min_{\{a_0, a_0, \cdots, a_{N-1}\}} G(\pi_0, a_0, \cdots, a_{N-1}) \quad (2.5)
\]

where

\[
G(\pi_0, a_0, \cdots, a_{N-1}) = \sum_{n=0}^{N-1} g(\pi_n, a_n) \quad (2.6)
\]

subject to the dynamics (2.1).

**A. Determination of the Transition Probability Matrices**

Let \( q_n^{(i)} \) denote the number of customers in \( Q_i \), including the one in service, just before the \( n \)th arrival and define \( X_{n+1}^{(i)} \) = potential number of departures from \( Q_i \) between arrival \( n \) and arrival \( n + 1 \), that is, the random variable that would count the number of service completions from \( Q_i \) during the interarrival time if the queue never emptied. The time between arrival \( n \) and arrival \( n + 1 \) is denoted \( A_{n+1} \). The state evolution of \( Q_i \) is given by a recursion of Lindley’s type (see [14]) as

\[
q_{n+1}^{(i)} = [q_n^{(i)} + 1(a_n = i) - X_{n+1}^{(i)}]^{+} \quad (2.7)
\]

where \( 1(A) \) is the indicator function of the event \( A \). Let

\[
q_j^{(i)} := P(X_{n+1}^{(i)} = j) = \int_{0}^{\infty} P(X_{n+1}^{(i)} = j | A_{n+1} = t) dF_A(t) = \int_{0}^{\infty} \mu^{(i)} e^{-\mu^{(i)} t} \frac{j!}{j!} dF_A(t) \quad (2.8)
\]

where the last step reflects the fact that the potential number of departures from an exponential server during the interval \( [0, t] \) has a Poisson distribution (see, e.g., [15]). The dependence of the transition probability matrix \( P_n^{(i)}(a_n) \) on action \( a_n \) is binary: either arrival \( n \) is routed to \( Q_i \); or it is routed elsewhere. To simplify the notation, denote

\[
P^{(i)}(a_n) = \begin{cases} 
[p_j^{(i)}(1)] & \text{if } a_n = i \\
[p_j^{(i)}(0)] & \text{otherwise} \end{cases} \quad (2.9)
\]

where

\[
P_{kl}^{(i)}(w) = \begin{cases} 
0 & \text{if } l > k + w \\
q_k^{(i)}(1) & \text{if } 0 < l \leq k + w \\
1 - \sum_{j=0}^{k-w} q_j^{(i)} & \text{if } l = 0 \end{cases} \quad (2.10)
\]

for \( w = 0, 1 \). Simple expressions of \( \{q_j^{(i)}\} \) can be obtained for a number of important distributions of the interarrival times, including deterministic, Erlangian, and uniform; and a redefinition of the elements of the transition probability matrices extends the treatment to exponential servers with state-dependent rates [16].

**B. A Remark on the Information Pattern, Cost Structure, and Computational Complexity**

Recall that \( X_n^{(i)} \) denotes the potential number of departures from \( Q_i \) during the interarrival time \( A_n \). The random variables \( X_n^{(i)}, X_n^{(j)} \) are not independent, because they both depend on \( A_n \). Therefore, the evolution of the number in \( Q_i \), namely \( \{q_n^{(i)}\} \), is coupled to the evolution of \( \{q_n^{(j)}\} \), the number in \( Q_j \).

Suppose that for computational purposes the maximum number in each queue is limited to \( L \), chosen large enough to render the truncation errors negligible. An array of dimension \((L + 1)^M\) is needed to store the joint probability distribution of the number in the \( M \) queues. For moderate to large values of \( M \), computing the joint probability density function (pdf) is hopeless. Fortunately, the information pattern and the cost structure considered in this paper make this computation unnecessary. Consider the information pattern first. No state observations are available to the decision maker (with the possible exception of the initial state). Therefore, for any given routing sequence \( \{a_n\} \) the marginal distributions of \( \{q_n^{(i)}\}, i = 1, \cdots M \) can be computed based on the prior statistics of the service times and interarrival distributions. (Recall that the interarrival times are i.i.d., and note in particular that they do not depend on the state of the system.) The cost structure is doubly additive. It is the sum, over the horizon of \( N \) arrivals, of the cost incurred at each arrival epoch. And the cost at each arrival epoch is the sum of the costs incurred by each individual queue. For the given information structure, the latter depends only on the marginal distributions. This decoupling of the evolution of the marginals is no longer valid if any state observation (in addition to the initial state) becomes available. And even if no measurements are made, the choice of a nonadditive cost structure renders the marginals insufficient to describe the evolution of the system. For instance, a cost that penalizes the probability that the total number in the system (i.e., in all queues) exceeds a certain threshold requires that the joint pdf be considered.

**III. Exhaustive Enumeration and Dynamic Programming Approaches**

The calculation of \( G(\pi_0, a_0, \cdots, a_{N-1}) \) for a given allocation sequence \( \{a_0, \cdots, a_{N-1}\} \) is, in principle, straightforward. Given that the number of admissible sequences is \( M^N \), however, a solution approach by exhaustive enumeration is unfeasible except for unrealistically small values of \( M \) and \( N \).

To avoid the geometric growth with \( N \), one is led to consider dynamic programming. At this point it is worth emphasizing that despite the intrinsic stochastic nature of the posed problem, we are dealing with the evolution of a deterministic system whose dynamics evolve according to (2.1). A control strategy for a deterministic system can be implemented either in an open-loop or in a closed-loop fashion. Dynamic programming yields closed-loop control laws.
of the form \(a_n = \gamma(n_a)\). Let \(J_n(\pi_n)\) be the minimum cost-to-go when \(N - n\) arrivals remain to be routed, and the state of the system is \(\pi_n\). Then, \(J_n(\pi_n)\) is recursively given by

\[
J_n(\pi_n) = 0
\]

(3.1a)

\[
J_n(\pi_n) = \min_{a \in A} \left\{ g(\pi_n, a) + J_{n+1}(\pi_n P(a)) \right\}
\]

(3.1b)

while the optimal control law \(\gamma(n_a)\) attains the minimum in (3.1b).

There are major difficulties with the numerical implementation of the algorithm defined by (3.1). Consider \(n_A\), in the first place, we would have to truncate the vector to its first \(K\) components. Suppose further that \(D\) values are used to discretize each component, and note that the specification of a point in the state space requires \(MK\) numbers. Therefore, the dimensionality of the discretized state space is of the order \(D^{MK}\). The above is actually an upper bound, because of the constraints imposed by \(\{\pi_n^{(i)}\}\) being distributions. Reasonable choices for \(D\) and \(K\), however, make the straightforward approach impractical, despite the fact that the dimensionality does not increase with \(N\). Alternatively, one can notice that to find the optimal routing sequence for a fixed value of \(\pi_0\), only the finite reachable set of states from \(\pi_0\) need be considered. This is certainly so due to the deterministic nature of the dynamics (2.1). Nevertheless, the number of reachable states grows geometrically fast with the stage, making the approach computationally intensive for even moderate values of \(M\) and \(N\), given that a globally optimal sequence is being sought.

Departing from the above approaches, we derive computationally efficient policy iteration algorithms based on a local optimization of actions. A formulation in terms of Pontryagin’s maximum principle is found to expose a useful structure in a clear form. Hence, we turn our attention to the latter approach.

While there is an extensive bibliography related to the application of Markovian decision process to the control of queues (see, e.g., [11], [12], [17], [18], and references therein), applications of Pontryagin’s maximum principle to queueing problems have been more limited and have focused on controlling continuous variables (service rates [19] or arrival rates [20]).

IV. APPROACH BASED ON THE MAXIMUM PRINCIPLE

We seek a routing sequence that minimizes \(G(x_0, a_0, \ldots, a_{N-1})\) subject to the system dynamics determined by (2.1). Pontryagin’s maximum principle (see, for instance, [21], [22], or [23]) views such an optimization as a joint choice of \(\{\pi_n, a_{n-1}\}, n = 0, 1, \ldots, N - 1\) subject to the constraints. To account for the constraints consider Lagrange multipliers \(\xi_n, n = 1, \ldots, N, \) and define

\[
\tilde{G}(\pi_0, a_0, \ldots, a_{N-1}) = G(\pi_0, a_0, \ldots, a_{N-1}) + \sum_{n=0}^{N-1} (\pi_n P(\pi_n) - \pi_{n+1}) \xi_{n+1}
\]

(4.1)

We can now define the Hamiltonian function

\[
H(\pi_n, \xi_{n+1}, a_n) = g(\pi_n, a_n) + \pi_n P(a_n) \xi_{n+1}
\]

(4.2)

Suppose that for an appropriate adjoint sequence \(\{\xi_n\}\) there exists an orbit \(\{\pi_n, a_{n-1}\}\) that minimizes (4.1). Pontryagin’s maximum principle establishes that \(\pi_n, a_{n-1}\) necessarily satisfy

\[
\xi_{n+1} = \frac{\partial H(\pi_n, \xi_{n+1}, a_n)}{\partial a_n} \left| \xi_n \right.,
\]

(4.3a)

\[
\xi_n = \frac{\partial H(\pi_n, \xi_{n+1}, a_n)}{\partial \pi_n} \left| \pi_n \right.,
\]

(4.3b)

\[
H(\pi_n, \xi_{n+1}, a_n) = \min_{a_n \in A} H(\pi_n, \xi_{n+1}, a_n).
\]

(4.3c)

Recall that \(\pi_n\) is a vector obtained by concatenating the \(M\) vectors \(\pi_n^{(i)}\), where \(\pi_n^{(i)}\) is the probability distribution of the number in \(Q_i\). Likewise, \(\xi_n\) decomposes in \(M\) vectors, each one associated with a particular queue: \(\xi_n^{(i)} = \left[\xi_n^{(i,1)} \cdots \xi_n^{(i,M)}\right]^T\). We also note that as a result of the linear nature of the cost structure (2.3)

\[
\frac{\partial g(\pi_n, a_n)}{\partial \pi_n} = \pi_n^{(i)}
\]

(4.4)

Therefore, the above necessary conditions specialize to

\[
\xi_{n+1}^{(i)} = \xi_n^{(i)} P^{(i)}(a_n)\]

(4.5a)

\[
\xi_n^{(i)} = \xi_n^{(i)} + P^{(i)}(a_n) \xi_{n+1}^{(i)}, \quad n = 0, \ldots, N - 1
\]

(4.5b1)

\[
\xi_n^{(i)} = 0
\]

(4.5b2)

\[
\bar{a}_n = \arg \min_{a_n \in A} \left\{ \sum_{i=1}^M \pi_n^{(i)} c_n^{(i)} + \sum_{i=1}^M \pi_n^{(i)} P^{(i)}(a_n) \xi_{n+1}^{(i)} \right\}
\]

(4.5c)

Regarding (4.5b2) we observe that in general, \(\xi_n^{(i)} = \frac{\partial K(\pi_n)}{\partial \pi_n^{(i)}}\), where \(K(\pi_n)\) is the terminal cost. In our formulation \(K(\cdot) = 0\). For future reference we define the myopic policy \(\{\bar{a}_0, \bar{a}_1, \ldots, \bar{a}_{N-1}\}\) as the sequence generated by the one step-ahead minimization

\[
\bar{a}_n = \arg \min_{a_n \in A} \left\{ \sum_{i=1}^M \pi_n^{(i)} c_n^{(i)} \right\}
\]

(4.6)

If the stage cost is the sojourn time of the arrival (2.4), the myopic policy coincides with the individually optimal policy [24], [17].

A. A Policy Iteration Algorithm

The necessary conditions for optimality dictated by the maximum principle suggest the following algorithm:

Algorithm 1:

Step 0) (Initialization) Set \(\bar{a}_0(\pi_0) = \bar{a}_n = \bar{a}_{n-1} = 0, n = 0, 1, \ldots, N - 1\).\n
Step 1) (Evaluation) For the current action sequence \(\bar{a}_n^{(k)}\) find the adjoint sequence \(\{\xi_n^{(k)}\}\) using (4.5b).

Step 2) (Policy Improvement) For every \(n, 0 \leq n \leq N - 1\), find the minimizing \(\bar{a}_n^{(k+1)}\) (4.5c) and the corresponding state \(\pi_n^{(k+1)}\) (4.5a). Resolve ties in favor of the old decision.

Step 3) (Test) Stop if \(\{\xi_n^{(k+1)}\} = \{\xi_n^{(k)}\}\). Otherwise, go back to step 1, now based on the new action sequence \(\bar{a}_n^{(k+1)}\).

B. Comments on the Algorithm

Define the scalars \(z_n^{(i)} := \pi_n^{(i)} c_n^{(i)}, i = 1, \ldots, M\). It follows from (4.5a) and (4.5b) that \(\xi_n^{(i)} = z_n^{(i)} - \pi_n^{(i)} c_n^{(i)}\), \(H(\pi_n, \xi_{n+1}, a_n) := x_n = x_n c_n\), and \(G(\pi_0, a_0, \ldots, a_{N-1}) = \pi_0 c_0\). The last expression highlights the two-point-boundary nature of the problem: \(x_0\) is the inner product of the initial state \(\pi_0\) and the adjoint vector \(\xi_0\). The latter has to satisfy recursion (4.5b1) with terminal condition (4.5b2). It is also worth observing that for a fixed action sequence, the evolution of both the adjoint vector and the state are linear.

We now proceed to establish a link to dynamic programming. To this goal, fix an arbitrary sequence \(\{\bar{a}_n\}\), recall the definition of \(z_n^{(i)}\), and note that the scalar \(x_n\), initialized as \(x_N = 0\), can be computed recursively as \(x_n = \sum_{i=1}^M \pi_n^{(i)} c_n^{(i)} + x_{n+1}\). It follows that \(x_n\) is the cost to be incurred during the remaining \(N - n\) arrival epochs, given the action sequence \(\{\bar{a}_j, j = 0, \ldots, N - 1\}\). Consider the policy improvement step in Algorithm 1. A change in action \(c_n^{(i)}\) does not affect \(\xi_{n+k}\), \(k > 1\), generated according
to recursion (4.5b). Hence, finding the minimizing $a_{n+1}^{(k+1)}$ in step 2 of Algorithm 1 yields a (local) improvement on the cost-to-go for fixed actions $\{a_i^{(k+1)}\}, l = 0, 1, \ldots, n$ and adjoints vectors $\{\xi_n^{(k)}\}$. The algorithm is similar to dynamic programming, except that the cost-to-go at each stage is based on the action sequence of the previous iteration, rather than on the optimal sequence for the remaining stages. The above local minimization structure emerges naturally from the maximum principle formulation, but is hidden by the dynamic programming formulation. The link with dynamic programming can be carried a step further by focusing on recursion (4.5b). In fact, the $j$th component of $\xi_n^{(k)}$ gives the contribution of $Q_j$ to the cost-to-go given that the current number in $Q_j$ is $j$, and that from this stage onwards the action sequence used to generate $\{\xi_n^{(k)}\}$ is followed. Whittle [23, pp. 245] examines the application of the maximum principle to stochastic systems. We have in fact taken what Whittle [23] calls the distributional approach by replacing the state variables by their distributions and considering the latter as deterministic state variables. Whittle [23], though, imposes the choice $\xi_n = \min_{a_n} G(p(a_n, a_{n-1}, \ldots, a_N))$. By insisting on this minimization, Whittle [23] derives the dynamic programming principle. Very reassuring indeed, but at a cost: "...it has never been demonstrated that this characterization has any computational or conceptual point" [23]. We focus on the local conditions for optimality and use them to derive an easily implementable algorithm.

We establish next the convergence of the algorithm.

C. Analysis of Convergence of the Policy Iteration Algorithm

Recall (2.6) and for a given action sequence $\{a_n\}$, recursively define the cost-to-go $G_{N-1}(\cdot)$

$$G_N(p_0, a_0, \ldots, a_{N-1}) = G(p_n, a_n, \ldots, a_{N-1})$$

(4.7a)

$$G_{N-1}(p_{n-1}, a_{n-1}, \ldots, a_{N-1}) = g(p_{n-1}, a_{n-1}, a_{N-1}) + G_{N-1}(p_{n-1}, p(a_{n-1}), a_{n-1}, \ldots, a_{N-1})$$

(4.7b)

where

$$G_{N-1}(p_{n-1}, a_{n-1}, \ldots, a_{N-1}) = \sum_{n=1}^{N-1} g(p_{n}, a_{n})$$

(4.8)

It follows that

$$G_{N-1}(p_{n-1}, a_{n-1}, \ldots, a_{N-1}) = H_{N-1}(p_{n-1}, \xi_{n-1}(a_{n-1}, \ldots, a_{N-1}), a_{n})$$

(4.9)

Lemma 1: For any fixed $p_i$, action sequence $\{a_i^{(k)}, \ldots, a_i^{(N)}\}$ and corresponding adjoint sequence $\{\xi_i^{(k)}, \ldots, \xi_i^{(N)}\}$, consider the local minimization

$$\bar{a}_i = \arg \min_{a_i \in A} \left\{ \sum_{i=1}^{M} a_i^{(i)} \cdot e_i^{(i)} + \sum_{j=1}^{M} z_j^{(i)} \cdot P^{(i)}(a_i) \cdot \xi_j^{(i)} \right\}$$

(4.10)

and impose the condition that ties are resolved in favor of $a_i^{(k)}$. Then

$$G_{N-1}(p_i, \bar{a}_i^{(k)}, \ldots, a_i^{(N)}) \leq G_{N-1}(p_i, a_i^{(k)}, \ldots, a_i^{(N)})$$

(4.11)

where equality holds if and only if $\bar{a}_i = a_i^{(k)}$.

Lemma 1 follows easily from the fact that

$$G_{N-1}(p_i, \bar{a}_i^{(k)}, \ldots, a_i^{(N)}) = \min_{a_i} H(p_i, \bar{a}_i^{(k)}, \ldots, a_i^{(N)}), a_i)$$

(4.12)

and neither $p_i$ nor $\xi_{i+1}$ depend on $a_i$.

Lemma 2: Let the sequence $\{a_n^{(k)}, k = 0, 1, \ldots\}$ be generated according to the prescription of Algorithm 1. Then,

$$G(p_0, a_0^{(k)}, \ldots, a_{N-1}^{(k)}) \leq G(p_0, a_0^{(k+1)}, \ldots, a_{N-1}^{(k+1)})$$

(4.13)

where equality holds if and only if $\{a_n^{(k)}\} \equiv \{a_n^{(k+1)}\}$, i.e., if the algorithm has converged.

Proof: The proof is by induction. According to recursion (4.7) and Lemma 1

$$G_N(p_0, a_0^{(k+1)}, \ldots, a_{N-1}^{(k+1)}) = g(p_0, a_0^{(k+1)}) + G_{N-1}(p_0, P(a_0^{(k+1)}), a_1^{(k+1)}, \ldots, a_{N-1}^{(k+1)})$$

(4.14)

Apply now Lemma 1 iteratively to $G_{N-1}(\cdot), l = 1, \ldots, N - 1$, and the proposition follows.

Theorem 1: Algorithm 1 converges in a finite number of steps to a control sequence that is not inferior to the initial sequence. Furthermore, the cost monotonically decreases with each iteration.

Proof: The monotonicity of the cost has been proved in Lemma 2. The convergence follows from the monotonicity and the fact that the policy space is finite.

D. Alternative Algorithms

Algorithm 1 converges, but not necessarily to the global minimum. Hence, it is of interest to consider alternative algorithms.

Reversing the Roles of $p_i$ and $\xi_i$: There is a certain symmetry between the roles played by the pdf of the number in the system $p_i$ and the adjoint vector $\xi_i$. At the beginning of iteration $k + 1$, Algorithm 1 finds the adjoint sequence $\{\xi_i^{(k)}\}$ induced by the sequence $\{a_i^{(k)}\}$. It then proceeds forward in time to find the new pairs $\{a_n^{(k+1)}, p_{n+1}^{(k+1)}, n = 0, \ldots, N - 1\}$. Algorithm 2 proceeds backward and reverses the roles of $p_i$ and $\xi_i$. Arguments similar to the ones used in Theorem 1 establish its convergence.

Looking at the Impact of $k$ Routing Decisions Simultaneously: Let $\Omega_1$ denote the set of all admissible routing sequences that satisfy (4.5a)-(4.5c). An action sequence $\gamma_k = \{a_0, \ldots, a_{N-1}\} \in \Omega_1$ is component-wise optimal in the sense that a change of a single decision cannot improve the overall cost. Step 2 in both Algorithms 1 and 2 seeks to improve the policy by altering one decision at a time. The deterministic nature of the system dynamics (2.1) allows us to write down its evolution every $k$ arrivals. This reformulated system, which we call the $k$-system, has the same structure of the original one, and both Algorithms 1 and 2 can be applied. Note, however, that at each policy improvement step the impact of all admissible values of a $k$-dimensional vector ($M^k$ possibilities), rather than the admissible values of scalar $a_k$ ($M$ possibilities), needs to be evaluated. While this clearly precludes the consideration of large values of $k$ (in the limit, with $k = N$, we are back to exhaustive enumeration), Theorem 2 points to the potential benefits of exploiting this structure.

Theorem 2: Algorithm 1 (or 2) applied to the $k$-system converges to a set $\Omega_k \subseteq \Omega_1$.

Proof: Formally, the system defined every $k$ arrivals does not differ from the original system. Hence, convergence of Algorithm 1 or 2 is guaranteed by Theorem 1 or 2. Now, at each policy improvement step the algorithm compares the $M^k$ possible routing decisions for $k$ consecutive arrivals. A policy $\gamma_k$ is a sample of the set $\Omega_k$ if no change in any $k$-dimensional decision vector improves the performance. These changes include the component by component changes that characterize $\Omega_k$.

Looking at the Impact of $k$ Decisions, but Accepting Only $m < k$: The variant proposed above arbitrarily divides the sequence in segments of length $k$. The algorithm could conceivably be trapped in
a local minimum that requires the simultaneous change of, say, two contiguous decisions that happen to belong to different segments. It is therefore clear that the set of local minima could further be reduced if any k contiguous changes could be made at every improvement step. This observation suggests yet another variant. At stage n in the policy improvement step, consider the next k contiguous decisions, and choose m < k. Accept only \(a_n, \ldots, a_{n+m-1}\), and move on to \(n+m\) (instead of accepting the whole vector and advancing to \(n+k\), as in the previous algorithm). This approach resembles somewhat a rolling-horizon technique.

V. A VARIATION IN THE INFORMATION PATTERN

We have so far considered the problem of routing \(N\) arrivals based only on the prior statistics of the system. Our approach yields a routing table with \(N\) entries. Suppose now that the arrival epochs become available and are used by the decision maker to compute the routing decision. Such a model, that precludes the a priori computation of a routing table, captures the scenario of remote locations with buffering capacities, but limited or costly communication capabilities, and a smart routing node. Let \(t_n, n = 0, 1, \ldots, N - 1\), denote the epoch of the \(n\)th arrival, and recall that \(\pi_n\) is the information state at epoch \(t_n, n = 0, \ldots, N - 1\). When the information about the arrival epochs was not available, we could compute the evolution of the information state just based on the initial distribution \(\pi_0\), the action sequence, and the description of the system [see (2.1)]. Now, the availability of the inter-arrival time \(T_n := t_n - t_{n-1}\) allows us to recalculate the information state at each arrival epoch as follows:

\[
\begin{align*}
\pi_{t_0} &= \pi_0, \\
\pi_{t_n}^{(i)} &= \pi_{t_{n-1}}^{(i)} P^{(i)}(a_{t_{n-1}}, T_n) \\
\end{align*}
\]  

(5.1a)

(5.1b)

where \(P^{(i)}(\cdot, t_n - t_{n-1})\) is as defined by (2.9)–(2.10), with

\[
\nu_j = \frac{(\mu^{(i)}_j T_n)^j}{j!} e^{-\mu^{(i)}_j T_n}.
\]  

(5.2)

At epoch \(t_n\) the decision maker can compute \(\pi_{t_n}\) and the probabilistic evolution of the information state \((\pi_{t_{n+1}}, \ldots, \pi_{t_{N-1}})\) for any interarrival time and control sequence, based on the statistics of the arrival process. Therefore, the optimal policy \(\gamma_{t_n}(\pi_{t_n})\) can in principle be computed using dynamic programming

\[
\begin{align*}
J_N(\pi_{t_N}) &= 0, \\
J_n(\pi_{t_n}) &= \min_{a \in A} \left\{ g(\pi_{t_n}, a) + \int_0^\infty J_{n+1}(\pi_{t_{n+1}} P(a, t)) dF_A(t) \right\}, \\
\gamma_{t_n}(\pi_{t_n}) &= \arg \min_{a \in A} \left\{ g(\pi_{t_n}, a) + \int_0^\infty J_{n+1}(\pi_{t_{n+1}} P(a, t)) dF_A(t) \right\}
\end{align*}
\]  

(5.3a)

(5.3b)

(5.4)

where the block diagonal components of matrix \(P(a, t)\) are defined by (2.9)–(2.10), (5.2). Compare these recursions with (3.1), and recall the discussion in Section III to motivate the need for an alternative algorithm.

Note that the dynamic programming recursions (5.3)–(5.4), yield the truly optimal closed-loop solution: At decision epoch \(t_n\) the controller not only uses the available time information, but also computes the cost-to-go anticipating that such information will also be available in the future [25]. The so-called open-loop feedback controller [11], [21], [25] represents a viable, suboptimal alternative. At each stage the open-loop feedback controller uses the current information to compute the optimal open-loop sequence and applies the first action of the sequence. The following stage brings fresh information, which means that the optimal open-loop sequence has to be recomputed, and so on. For the problem at hand, the algorithm can be summarized as follows.

At epochs \(t_n, n = 0, 1, \ldots, N - 1\) do the following:

1. **Phase 1** Compute the information state \(\pi_{t_n}\) according to (5.1).

2. **Phase 2** Compute the sequence \(\{a_{t_0}^{(n)}, a_{t_1}^{(n)}, \ldots, a_{t_{N-1}}^{(n)}\}\) according to Algorithm 1 (or one of its variants).

3. **Phase 3** Set \(a_{t_n} = a_{t_n}^{(n)}\), and discard the rest of the above sequence. Keep in memory \(t_n, \pi_{t_n}, a_{t_n}\).

VI. NUMERICAL RESULTS

Consider a system of three single server queues with service rates \((\mu_1^{(1)}, \mu_2^{(1)}, \mu_3^{(1)}) = (1, 1, 2)\). The initial number in the system is \((q_0^{(1)}, q_0^{(2)}, q_0^{(3)}) = (10, 5, 0)\), the arrival rate \(\lambda = 2\), and the horizon \(N = 30\) arrivals. The cost is the sum of the expected sojourn times of the arrivals. Table I depicts the evolution of the cost as a function of the number of iterations for both Algorithms 1 and 2, under two different initializations: the myopic policy and the round robin scheduling. Algorithm 2 initialized with the myopic strategy achieves a cost of 44,738, slightly lower than the 44,756 cost achieved by Algorithm 1 initialized with the round robin policy. In all four cases the algorithms converge in a few iterations, ranging from one to four, and the cost decreases with each iteration. The near-optimality of the myopic policy, while not uncommon, is not a general rule, as will be shown in the sequel. To ameliorate the problem of local minima, we considered a variant of the basic Algorithm 1 and 2, constructed in the spirit of the discussions in Section IV-D2). The variant of Algorithm 1 that looks two steps ahead, but accepts only the first one, is called Algorithm 1a. These and a number of experiments reported elsewhere [16] suggest that the various algorithms converge to local minima that are not far from the global minimum. A plausible explanation for the mechanism at work is the following. Call allocation can be thought of as consisting of two processes: the allocation of workload to each queue (i.e., determination of the number of arrivals to be sent to each queue) and sequencing (i.e., determination of the order in which the arrivals are to be routed). Gross deviations of the workload allocation from the optimal one should translate into considerably higher costs. Therefore, it is reasonable to expect that all stationary points of the algorithms yield allocations that are close to the optimal. Fig. 1 provides experimental evidence to support this conjecture. The following parameters were chosen: arrival rate \(\lambda = 3\), service rates \(\mu_1^{(1)} = \mu_2^{(1)} = 1, \mu_3^{(1)} = 2\), initial state \((5, 0, 5)\). The two sets of three plots on the upper side of the figure display the histograms of the cost for the initial policy, and the policies corresponding to Algorithm 1 and Algorithm 1a. Both sets display the same data, but on different scales. Algorithm 1 does well, but it is outperformed by Algorithm 1a. The three triplets display the histograms of the allocation of arrivals to \(Q_1, Q_2,\) and \(Q_3\) for the randomly chosen initial policy, the policy generated by Algorithm 1, and the policy generated by Algorithm 1a. The distribution of the allocation is striking:
in all cases, for both policies, \( Q_3 \) was sent 11 arrivals, in sharp contrast with the wide distribution exhibited by the initial (random) sequence. Of the remaining 10 arrivals, \( Q_1 \) received either two or three (and consequently, \( Q_2 \) got either eight or seven). Therefore, the total workload to be processed by \( Q_1, Q_2 \), and \( Q_3 \) is either \((5+2)/1, (0+8)/1, (5+11)/2\), or \((5+3)/1, (0+7)/1, (5+11)/2\), respectively. (We do not suggest that equating the workload is always optimal; only that it works for this example.) The implication is that, at least in this example, both algorithms do a remarkable job in finding the right allocation. The fact that Algorithm 1a outperforms Algorithm 1 has to be attributed to a better choice of the sequence that implements the allocation. The random choice of the initial sequence was used to test the algorithms. For this example, a far better initial choice is the individually optimal sequence. For the chosen parameters, the individually optimal sequence yields a cost of 48.2324, with an allocation of \((2, 7, 12)\). Algorithm 1 so initialized converges to a sequence with allocation \((2, 8, 11)\) and cost 48.0658. The sequence generated by Algorithm 1a has the same allocation \((2, 8, 11)\), but does a better job at sequencing, yielding a lower cost of 47.9043. This tends to reinforce the conjecture that the set of policies yielded by the algorithms are nearly optimal from the point of view of workload allocation.

The examples discussed so far may induce the notion that the myopic policy is always near optimal, and consequently, the algorithms developed here are of little practical impact.

To check the validity of such a notion it is worth looking into cost criteria different from average sojourn time. One alternative criterion, considered in the results presented in Fig. 2, is to penalize the probability that the mean delay seen by an arrival exceeds a certain threshold. The rationale is that beyond certain threshold impatient customers may abandon. This may well model human behavior, e.g., in the case of 1–800 calls, or a protocol mechanism with time-out (see [14]). The following parameters were used in the experiment: arrival rate \( \lambda = 6 \), service times \( \mu^{(1)} = 1, \mu^{(2)} = 2, \mu^{(3)} = 3 \). horizon = 21 arrivals, initial state \((5, 10, 30)\). Algorithm 1, initialized with the myopic policy, achieves a dramatic reduction of the cost in only two iterations. The interpretation of these results is not difficult. The system is initially loaded. In fact, the initial expected delays faced by the first arrival are \((5, 5, 10)\), depending on the routing decision. Compare this to the cost, which is the probability that the expected waiting time exceeds a delay of five units. The fact that \( \lambda = \mu_1 + \mu_2 + \mu_3 \) implies that the congestion is unlikely to be cleared for the duration of the horizon. The individually optimal policy chooses the best decision for each arrival, regardless of its impact on other arrivals. The policy chosen by Algorithm 1 sacrifices the first eight arrivals, which are likely to suffer long delays, to clear the way for the remaining 13 arrivals. By doing so, it achieves a 25% reduction from the cost of the individually optimal policy. Balancing the workload is not a good idea in this example. Indeed, the individually optimal policy, which performs poorly, yields a more balanced load than that generated by Algorithm 1. Should blocking have been included in the admissible action set, and should the cost of blocking not have been excessive, Algorithm 1 would have considered it.

VII. CONCLUSION

We have presented an approach to the problem of routing \( N \) arrivals to \( M \) parallel queues based on Pontryagin’s maximum principle. Two policy iteration algorithms have been proposed, and their convergence established. Furthermore, the cost decreases mono-
Fig. 2. Individually optimal policy not necessarily near-optimal.

tonically with each iteration. Variations of the basic algorithms have been proposed, including one that covers a richer information pattern. Extensive numerical experiments point to the near optimality of the algorithms. Finally, we showed that the myopic policy is not necessarily a good policy. Future lines of research extend in three directions:

i) The optimal stationary open-loop policy for the infinite horizon problem;

ii) The optimal stationary policy for the infinite horizon problem, when measurements are made available every $N$ arrivals;

iii) The multiclass environment, with class dependent service time requirements.

ACKNOWLEDGMENT

Discussions with S. Durntovic-Johiri, Y. Kogan, and Y. Levy from AT&T Bell Laboratories are greatly appreciated. We are grateful to Prof. M. Neuts for his comments on an early version of the paper.

REFERENCES


Design of a $t^2$-Optimal Regulator: The Limits of Performance Approach

Y. T. Teo and T. T. Tay

Abstract—In this paper, the design of a $t^2$-optimal regulator using the limits of performance [1] approach to constrain the magnitude of the control signal is proposed. We show that for the $t^2$-optimal control case, performance indexes based on a weighted sum or weighted maximum of the process output and control output fail to achieve the objective of compromising between tight control and control effort. The limits of performance curve, which is a plot of the best achievable specifications with a given system and control configuration, on the other hand allows the designer to achieve appropriate compromise. We further show that the limits of performance curve is formed by a finite number of linear equations and its gradient is monotonically nondecreasing. A systematic method to construct the limits of performance curve is proposed.

I. INTRODUCTION

For certain practical problems, such as the regulation of the read/write head of a disk drive, the control objective is to keep the magnitude of the error signals to within a certain tolerance while