A GENERALIZED 'SURROGATE PROBLEM' METHODOLOGY FOR ON-LINE STOCHASTIC DISCRETE OPTIMIZATION*

Kagan Gokbayrak
Department of Manufacturing Engineering
Boston University
Boston, MA 02215
kgokbayr@bu.edu

Christos G. Cassandras
Department of Manufacturing Engineering
Boston University
Boston, MA 02215
cgc@bu.edu

Submitted to JOTA
January 2001

Abstract

We consider stochastic discrete optimization problems where the decision variables are non-negative integers and propose a generalized "surrogate problem" methodology that modifies and extends previous work in [1]. Our approach is based on an *on-line* control scheme which transforms the problem into a "surrogate" continuous optimization problem and proceeds to solve the latter using standard gradient-based approaches while simultaneously updating both actual and surrogate system states. In contrast to [1], the proposed methodology applies to arbitrary constraint sets. It is shown that, under certain conditions, the solution of the original problem is recovered from the optimal surrogate state. Applications of this approach include solutions to multicommodity resource allocation problems, where, exploiting the convergence speed of the method, one can overcome the obstacle posed by the presence of local optima.

*This work was supported in part by the National Science Foundation under Grants EEC-95-27422 and ACI-98-73339, by AFOSR under contract F49620-98-1-0387, by the Air Force Research Laboratory under contract F30602-99-C-0057 and by EPRI/ARO under contract WO8333-03.

1 Introduction

We consider stochastic discrete optimization problems where the decision variables are non-negative integers. Problems of this type abound, for instance, in manufacturing systems and communication networks. In a manufacturing system setting, examples include the classic buffer allocation problem, where K buffer slots are to be distributed over N manufacturing workstations so as to optimize performance criteria involving throughput or mean system time; a variant of this problem involving the use of kanban (rather than buffer slots) to be allocated to different workstations [2]; and determining the optimal lot size for each of N different part types sharing resources in a production facility with setup delays incurred when a switch from a lot of one part type to another occurs [3]. In a communication network setting, similar buffer allocation issues arise, as well as transmission scheduling problems where a fixed number of time slots forming a "frame" must be allocated over several call types [4]. Such optimization problems are also very common in any discrete resource allocation setting [5], as well as in control policies for Discrete Event Systems (DES) that are parameterized by discrete variables such as *thresholds* or *hedging points.*

The optimization problem we are interested in is of the general form

$$\min_{r \in A_d} J_d(r) = E[L_d(r,\omega)] \tag{1}$$

where $r \in \mathbb{Z}_+^N$ is a decision vector or "state" and A_d represents a constraint set. In a stochastic setting, let $L_d(r,\omega)$ be the cost incurred over a specific sample path ω when the state is r and $J_d(r) = E[L_d(r,\omega)]$ be the expected cost of the system operating under r. The sample space is $\Omega = [0,1]^\infty$, that is, $\omega \in \Omega$ is a sequence of random numbers from $[0,1]$ used to generate a sample path of the system. The cost functions are defined as $L_d : A_d \times \Omega \to \mathbb{R}$ and $J_d : A_d \to \mathbb{R}$, and the expectation is defined with respect to a probability space $(\Omega, \Im, P)$ where $\Im$ is an appropriately defined σ-field on Ω and P is a conveniently chosen probability measure. In the sequel, 'ω' is dropped from $L_d(r,\omega)$ and, unless otherwise noted, all costs will be over the same sample path.

The problem (1) is a notoriously hard stochastic integer programming problem. Even in a deterministic setting, where $J_d(r) = L_d(r)$, this class of problems is NP-hard (see [5], [6] and references therein). In some cases, depending upon the form of the objective function $J_d(r)$ (e.g., separability, convexity), efficient algorithms based on finite-stage dynamic programming or generalized Lagrange relaxation methods are known (see [5] for a comprehensive discussion on aspects of deterministic resource allocation algorithms). Alternatively, if no a priori information is known about the structure of the problem, then some form of a search algorithm is employed (e.g., Simulated Annealing [7], Genetic Algorithms [8]).

When the system operates in a stochastic environment (e.g., in a resource allocation setting, users request resources at random time instants or hold a particular resource for a random period of time) and no closed-form expression for $E[L_d(r)]$ is available, the problem is further complicated by the need to estimate $E[L_d(r)]$. This generally requires Monte Carlo simulation or direct measurements made on the actual system. Most known approaches are based on some form of random search, as in algorithms proposed by Yan and Mukai [9], Gong et al [10], Shi and Olafsson [11]. Another recent contribution to this area involves the *ordinal* optimization approach presented in [12] and used in [13] to solve a class of resource allocation problems. Even though the approach in [13] yields a fast algorithm, it is still constrained to iterate so that every step involves the transfer of no more than a single resource from one user to some other user. One can expect, however, that much faster improvements can be realized in a scheme allowed to reallocate multiple resources from users whose cost-sensitivities are small to users whose sensitivities are much larger. This is precisely the rationale of most gradient-based continuous optimization schemes, where the gradient is a measure of this sensitivity.

With this motivation in mind, a new approach was proposed in [1] based on the following idea: The *discrete* optimization problem (1) is transformed into a "surrogate" *continuous* optimization problem which is solved using standard gradient-based methods; its solution is then transformed back into a solution of the original problem. Moreover, this process is designed explicitly for *on-line* operation. That is, at every iteration step in the solution of the surrogate continuous optimization problem, the surrogate continuous state is immediately transformed into a feasible discrete state r. This is crucial, since whatever information is used to drive the process (e.g., sensitivity estimates) can only be obtained from a sample path of the *actual* system operating under r. It was shown in [1] that for resource allocation problems, where the constraint set is of the form $A_d = \left\{ r : \sum_{i=1}^{N} r_i = K \right\}$, the solution of (1) can be recovered from the solution of the surrogate continuous optimization problem; the latter is obtained using a stochastic approximation algorithm which converges under standard technical conditions.

The contributions of this paper are the following. First, we generalize the methodology presented in [1] to problems of the form (1) which are not necessarily limited to constraints such as $A_d = \left\{ r : \sum_{i=1}^{N} r_i = K \right\}$, including the possibility of unconstrained problems. Second, we modify the approach developed in [1] in order to improve its computational efficiency. In particular, computational efficiency is gained in the following respects:

1. A crucial aspect of the "surrogate problem" method is the fact that the surrogate state, denoted by $\rho \in \mathbb{R}_+^N$, can be expressed as a convex combination of at most $N + 1$ points in A_d, where N is the dimensionality of $r \in A_d$. Determining such points is not a simple task. In [1], this was handled using the Simplex Method of Linear Programming, which can become inefficient for large values of N. In this paper, we show that for any surrogate state ρ, a selection set $S(\rho)$ of such $N + 1$ points, *not necessarily in* A_d, can be identified through a simple algorithm of linear complexity. Moreover, this algorithm applies to any problem of the form (1), not limited to any special type of constraint set A_d.

2. In solving the surrogate continuous optimization problem, a surrogate objective function is defined whose gradient is estimated in order to drive a stochastic approximation type of algorithm. The gradient estimate computation in [1] involves the inversion of an $N \times N$ matrix. In this paper, we show that this is not needed if one makes use of the selection set $S(\rho)$ mentioned above, and the gradient estimate computation is greatly simplified.

The price to pay for the generalization of the approach is the difficulty in establishing a general result regarding the recovery of the solution of (1) from the solution of the surrogate problem as was done in our earlier work [1]. We are able, however, to still do so for two interesting cases.

Despite this difficulty, the empirical evidence to date indicates that this generalized methodology provides the optimal solutions under appropriate technical conditions guaranteeing convergence of a stochastic approximation scheme.

A third contribution of this paper is in tackling a class of particularly hard *multicommodity* discrete optimization problems, where multiple local optima typically exist. Exploiting the convergence speed of the surrogate method, we present, as an application of the proposed approach, a systematic means for solving such combinatorially hard problems.

The rest of the paper is organized as follows. In Section 2, we give an overview of our basic approach. In Section 3, we present the key results enabling us to transform a discrete stochastic optimization problem into a "surrogate" continuous optimization problem. In Section 4, we discuss the construction of appropriate "surrogate" cost functions for our approach and the evaluation of their gradients. Section 5 discusses how to recover the solution of the original problem from that of the "surrogate" problem. Section 7 contains some numerical examples and applications and describes how the "surrogate problem" method is used to solve multicommodity resource allocation problems that exhibit multiple local optima.

2 Basic approach for on-line control

In the sequel, we shall adopt the following notational conventions as in [1]. We shall use subscripts to indicate components of a vector (e.g., r_i is the ith component of r). We shall use superscripts to index vectors belonging to a particular set (e.g., r^j is the jth vector within a subset of $\mathbb{Z}_+^N$ that contains such vectors). Finally, we reserve the index n as a subscript that denotes iteration steps and not vector components (e.g., r_n is the value of r at the nth step of an iterative scheme, not the nth component of r).

The expected cost function $J_d(r)$ is generally nonlinear in r, a vector of integer-valued decision variables, therefore (1) is a nonlinear integer programming problem. One common method for solving this problem is to relax the integer constraints on all r_i so that they can be regarded as continuous (real-valued) variables and then to apply standard optimization techniques such as gradient-based algorithms. Let the "relaxed" set A_c contain the original constraint set A_d and define $\bar{L}_c : \mathbb{R}_+^N \times \Omega \to \mathbb{R}$ to be the cost function over a specific sample path. As before let us drop 'ω' from $\bar{L}_c(\rho, \omega)$ and agree that unless otherwise noted all costs will be over the same sample path. The resulting "surrogate" problem then becomes: Find ρ^* that minimizes the "surrogate" expected cost function $J_c : \mathbb{R}_+^N \to \mathbb{R}$ over the continuous set A_c, i.e.,

$$J_c(\rho^*) = \min_{\rho \in A_c} J_c(\rho) = E[\bar{L}_c(\rho)] \tag{2}$$

where $\rho \in \mathbb{R}_+^N$, is a real-valued state, and the expectation is defined on the same probability space $(\Omega, \Im, P)$ as described earlier. Assuming an optimal solution ρ^* can be determined, this state must then be mapped back into a discrete vector by some means (usually, some form of truncation). Even if the final outcome of this process can recover the actual r^* in (1), this approach is strictly limited to *off-line* analysis: When an iterative scheme is used to solve the problem in (2) (as is usually the case except for very simple problems of limited interest), a sequence of points $\{\rho_n\}$ is generated; these points are generally continuous states in A_c, hence they may be infeasible in the original discrete optimization problem. Moreover, if one has to estimate $E[\bar{L}_c(\rho)]$ or $\frac{\partial E[\bar{L}_c(\rho)]}{\partial \rho}$ through simulation, then a simulation model of the surrogate problem must be created, which is also not generally feasible. If, on the other hand, the only cost information available is through direct observation of sample paths of an actual system, then there is no obvious way to estimate $E[\bar{L}_c(\rho)]$ or $\frac{\partial E[\bar{L}_c(\rho)]}{\partial \rho}$, since this applies to the real-valued state ρ, not to the integer-valued actual state r.

As in [1], we adopt here a different approach intended to operate *on line*. In particular, we still invoke a relaxation such as the one above, i.e., we formulate a surrogate continuous optimization problem with some state space $A_c \subset \mathbb{R}_+^N$ and $A_d \subset A_c$. However, at every step n of the iteration scheme involved in solving the problem, both the continuous and the discrete states are simultaneously updated through a mapping of the form $r_n = f_n(\rho_n)$. This has two advantages: First, the cost of the original system is continuously adjusted (in contrast to an adjustment that would only be possible at the end of the surrogate minimization process); and second, it allows us to make use of information typically needed to obtain cost sensitivities from the actual operating system at every step of the process.

The basic scheme we consider is the same as in [1] and is outlined below for the sake of self-sufficiency of the paper. Initially, we set the "surrogate system" state to be that of the actual system state, i.e.,

$$\rho_0 = r_0 \tag{3}$$

Subsequently, at the nth step of the process, let $H_n(\rho_n, r_n, \omega_n)$ denote an estimate of the sensitivity of the cost $J_c(\rho_n)$ with respect to ρ_n obtained over a sample path ω_n of the actual system operating under allocation r_n; details regarding this sensitivity estimate will be provided later in the paper. Two sequential operations are then performed at the nth step:

1. The continuous state ρ_n is updated through

$$\rho_{n+1} = \pi_{n+1}[\rho_n - \eta_n H_n(\rho_n, r_n, \omega_n)] \tag{4}$$

   where $\pi_{n+1} : \mathbb{R}^N \to A_c$ is a projection function so that $\rho_{n+1} \in A_c$ and η_n is a "step size" parameter.

2. The newly determined state of the surrogate system, ρ_{n+1}, is transformed into an actual feasible discrete state of the original system through

$$r_{n+1} = f_{n+1}(\rho_{n+1}) \tag{5}$$

   where $f_{n+1} : A_c \to A_d$ is a mapping of feasible continuous states to feasible discrete states which must be appropriately selected as will be discussed later.

One can recognize in (4) the form of a stochastic approximation algorithm (e.g., [14]) that generates a sequence $\{\rho_n\}$ aimed at solving (2). However, there is an additional operation (5) for generating a sequence $\{r_n\}$ which we would like to see converge to r^* in (1). It is important to note that $\{r_n\}$ corresponds to feasible realizable states based on which one can evaluate estimates

$H_n(\rho_n, r_n, \omega_n)$ from observable data, i.e., a sample path of the actual system under r_n (not the surrogate state ρ_n). We can therefore see that this scheme is intended to combine the advantages of a stochastic approximation type of algorithm with the ability to obtain sensitivity estimates with respect to discrete decision variables. In particular, sensitivity estimation methods for discrete parameters based on Perturbation Analysis (PA) and Concurrent Estimation [15],[16] are ideally suited to meet this objective.

Before addressing the issue of obtaining estimates $H_n(\rho_n, r_n, \omega_n)$ necessary for the optimization scheme described above to work, there are two other crucial issues that form the cornerstones of the proposed approach. First, the selection of the mapping f_{n+1} in (5) must be specified. Second, a surrogate cost function $\bar{L}_c(\rho, \omega)$ must be identified and its relationship to the actual cost $L_d(r, \omega)$ must be made explicit. These issues are discussed next, in Sections 3 and 4 respectively for the problem (1), which, as previously mentioned, is not limited to the class of resource allocation problems considered in our earlier work [1].

3 Continuous-to-discrete state transformations

Let us first define $\mathcal{C}(\rho)$, the set of vertices of the unit "cube" around the surrogate state as

$$\mathcal{C}(\rho) = \{r | \forall i \; r_i \in \{\lfloor \rho_i \rfloor, \; \lceil \rho_i \rceil\}\}$$

where, for any $x \in \mathbb{R}$, $\lceil x \rceil$ and $\lfloor x \rfloor$ denote the ceiling (smallest integer $\geq x$) and floor (largest integer $\leq x$) of x respectively. Note that when $\rho_i \in \mathbb{Z}$, all the ith components of the cube elements are the same ($= \rho_i$) decreasing the dimension of the cube by one. In order to avoid the technical complications due to integer components in ρ, let us agree that whenever this is the case we will perturb the integer components to obtain a new state $\hat{\rho}$ whose components are non-integer, and then relabel this state as ρ.

Next, we define $\mathcal{N}(\rho)$, the set of all *feasible neighboring discrete states* in $\mathcal{C}(\rho)$ as:

$$\mathcal{N}(\rho) = \mathcal{C}(\rho) \cap A_d \tag{6}$$

A more explicit and convenient characterization of the set $\mathcal{N}(\rho)$ is

$$\mathcal{N}(\rho) = \{r | r = \lfloor \rho \rfloor + \tilde{r} \text{ for all } \tilde{r} \in \{0, 1\}^N\} \cap A_d$$

where $\lfloor \rho \rfloor$ is the vector whose components are $\lfloor \rho \rfloor_i = \lfloor \rho_i \rfloor$. In other words, $\mathcal{N}(\rho)$ is the set of vertices of the unit "cube" containing ρ that are in the feasible discrete set A_d.

In earlier work (see [1]), we limited ourselves to resource allocation problems with linear capacity constraints. For this class of problems, we used $A_c = conv(A_d)$ as the feasible set in the "surrogate" continuous state space. When the feasible set A_d is not a polyhedron, the set $A_c = conv(A_d)$ may include discrete states that are not in A_d. In order to prevent this and to generalize the approach, we modify the definition of A_c, given the set $\mathcal{N}(\rho)$, as follows:

$$A_c = \bigcup_{\rho \in \mathbb{R}_+^N} conv(\mathcal{N}(\rho)) \tag{7}$$

Note that $A_c \subseteq conv(A_d)$ is the union of the convex hulls of feasible discrete points in every cube, and is not necessarily convex. Note also that the definition reduces to $A_c = conv(A_d)$ when A_d is formed by all the discrete points in a polyhedron.

Now we are ready to define the set of *transformation functions* $\mathcal{F}_\rho$ as follows:

$$\mathcal{F}_\rho = \{f | f : A_c \to A_d, \ \ (f(\rho))_i \in \{\lceil \rho_i \rceil, \lfloor \rho_i \rfloor\}, \ i = 1, \ldots, N\}$$

The purpose of $f \in \mathcal{F}_\rho$ is to transform some continuous state vector $\rho \in A_c$ into a "neighboring" discrete state vector $r \in \mathcal{N}(\rho)$ obtained by seeking $\lceil \rho_i \rceil$ or $\lfloor \rho_i \rfloor$ for each component $i = 1, \ldots, N$. The existence of such a transformation is guaranteed by the projection mapping π in (4), which ensures that $\rho \in A_c$, therefore $\mathcal{N}(\rho)$ is non-empty. A convenient element of $\mathcal{F}_\rho$ that we shall use throughout the paper is

$$f(\rho) = \arg \min_{r \in \mathcal{N}(\rho)} \|\rho - r\|$$

which maps the surrogate state ρ to the closest feasible neighbor in $\mathcal{N}(\rho)$. However, our analysis is certainly not limited to this choice.

A key element of our approach is based on the fact that ρ can be expressed as a convex combination of *at most* $N + 1$ points in $\mathcal{C}(\rho)$, as shown in Theorem 3.1 below. Given that the cardinality of $\mathcal{C}(\rho)$ is combinatorially explosive, i.e., 2^N, determining the set of these points is not a simple task. In [1], it was shown that such a set of feasible points, $\mathcal{N}_N(\rho)$, a subset of $\mathcal{N}(\rho)$, can be determined using the Simplex Method when problems of the form (1) are limited to constraint sets $A_d = \left\{ r : \sum_{i=1}^N r_i = K \right\}$. In what follows, we provide a different approach based on defining a *selection set* $\mathcal{S}(\rho)$ which (a) allows us to specify the $N+1$ points in $\mathcal{C}(\rho)$ that define a set whose convex hull includes ρ for problems with arbitrary A_d, (b) is much simpler than the Simplex Method, and (c) simplifies the gradient estimation procedure as we will see in Section 4. An important distinction between $\mathcal{N}_N(\rho)$ and $\mathcal{S}(\rho)$ is that the latter is not limited to include only feasible points $r \in \mathcal{N}(\rho)$.

Definition 3.1 *The set $\mathcal{S}(\rho) \subseteq \mathcal{C}(\rho)$ is a selection set if it satisfies the following conditions:*

- $|\mathcal{S}(\rho)| = N + 1$
- The surrogate state ρ resides in the convex hull of $\mathcal{S}(\rho)$, i.e., there exists $\{\alpha_i\}$ such that

$$\rho = \sum_{i=0}^{N} \alpha_i r^i, \quad \text{with} \quad \sum_{i=0}^{N} \alpha_i = 1, \quad \alpha_i \geq 0, \quad r^i \in \mathcal{S}(\rho)$$

- The vectors in the set $\left\{ \bar{r}^i | \bar{r}^i = \left[\begin{array}{cc} 1 & r^i \end{array} \right], \ \ r^i \in \mathcal{S}(\rho) \right\}$ are linearly independent.

Next we will show the existence of the selection set $\mathcal{S}(\rho)$ by a constructive proof.

Theorem 3.1 *A selection set $\mathcal{S}(\rho)$ exists for any $\rho \in \mathbb{R}_+^N$.*

Proof. We construct a selection set $\mathcal{S}(\rho)$ for $\rho = [\rho_1, ..., \rho_N]$ as described below and prove that it satisfies all three conditions in Definition 3.1.

Let us define e_i as the N-dimensional unit vector whose ith component is 1; the *residual vector* $\tilde{\rho} = \rho - \lfloor \rho \rfloor$; and the N-dimensional ordering vector o such that $o_k \in \{1, ..., N\}$, $k = 1, ..., N$, and o_k satisfies

$$\tilde{\rho}_{o_k} \leq \tilde{\rho}_{o_{k+1}} \text{ for } k = 1, ..., N-1$$

Note that the definition of $\tilde{\rho}$ implies that $0 < \tilde{\rho}_j < 1$ for all $j = 1, ..., N$. Next, we define

$$\tilde{r}^{o_l} = \sum_{k=l}^{N} e_{o_k} \tag{8}$$

and

$$\alpha_{o_l} = \begin{cases} \tilde{\rho}_{o_l} - \tilde{\rho}_{o_{l-1}} & l > 1 \\ \tilde{\rho}_{o_1} & l = 1 \end{cases} \geq 0 \tag{9}$$

It follows from (9) that we can write

$$\tilde{\rho}_{o_l} = \sum_{k=1}^{l} \alpha_{o_k} \tag{10}$$

and note that

$$\tilde{\rho}_{o_N} = \sum_{k=1}^{N} \alpha_{o_k} = \sum_{j=1}^{N} \alpha_j$$

Using (9) we have defined α_i for $i = 1, \ldots, N$. In addition, we now define

$$\alpha_0 = 1 - \sum_{i=1}^{N} \alpha_i = 1 - \tilde{\rho}_{o_N} > 0 \tag{11}$$

Similarly, (8) defines $\tilde{r}^i$ for $i = 1, \ldots, N$. In addition, we define

$$\tilde{r}^0 = \mathbf{0} \tag{12}$$

where $\mathbf{0} = [0...0]$ is the N-dimensional zero vector. Note that we can write

$$\tilde{\rho} = \sum_{l=1}^{N} \tilde{\rho}_{o_l} e_{o_l} \tag{13}$$

Combining (10) and (13) and changing the summation indices gives

$$\tilde{\rho} = \sum_{l=1}^{N} \sum_{k=1}^{l} \alpha_{o_k} e_{o_l} = \sum_{k=1}^{N} \sum_{l=k}^{N} \alpha_{o_k} e_{o_l} \tag{14}$$

Then, using (8) we get

$$\tilde{\rho} = \sum_{k=1}^{N} \alpha_{o_k} \tilde{r}^{o_k} = \sum_{j=1}^{N} \alpha_j \tilde{r}^j \tag{15}$$

Next, we define

$$r^j = \lfloor \rho \rfloor + \tilde{r}^j \tag{16}$$

Then, we can write

$$\begin{aligned}\sum_{j=0}^{N} \alpha_j r^j &= \sum_{j=0}^{N} \alpha_j \lfloor \rho \rfloor + \sum_{j=0}^{N} \alpha_j \tilde{r}^j \\ &= \lfloor \rho \rfloor \sum_{j=0}^{N} \alpha_j + \sum_{j=0}^{N} \alpha_j \tilde{r}^j\end{aligned}$$

Observing that $\sum_{j=0}^{N} \alpha_j = 1$ from (11), and that

$$\sum_{j=0}^{N} \alpha_j \tilde{r}^j = \sum_{j=1}^{N} \alpha_j \tilde{r}^j + \alpha_0 \tilde{r}^0 = \tilde{\rho}$$

from (12) and (15), it follows that

$$\sum_{j=0}^{N} \alpha_j r^j = \lfloor \rho \rfloor + \tilde{\rho} = \rho \tag{17}$$

i.e., the convex hull formed by $\mathcal{S}(\rho) = \{r^0, ..., r^N\}$, with r^i defined in (16), contains ρ. This satisfies the second condition in Definition 3.1. Moreover, from (8), (12), and (16), it is obvious that $|\mathcal{S}(\rho)| = N + 1$, satisfying the first condition as well.

It remains to show that the vectors $\left[\begin{array}{cc} 1 & r^i \end{array}\right]$ with r^i defined in (16) are linearly independent. Consider the matrix $\left[\begin{array}{cc} e & \mathbf{R} \end{array}\right]$ where $e = [1 \cdots 1]'$ is the $(N+1)-$ dimensional vector of $1'$s and $\mathbf{R}$ is the matrix whose rows are vectors from $\mathcal{S}(\rho)$ such that

$$\mathbf{R} = \begin{bmatrix} r^0 \\ r^{o_N} \\ \vdots \\ r^{o_1} \end{bmatrix}$$

Using (8), (12), and (16), one can write $\left[\begin{array}{cc} 1 & r^{o_l} \end{array}\right] - \left[\begin{array}{cc} 1 & r^{o_{l+1}} \end{array}\right] = \left[\begin{array}{cc} 0 & e_{o_l} \end{array}\right]$ for $l < N$ and $\left[\begin{array}{cc} 1 & r^{o_N} \end{array}\right] - \left[\begin{array}{cc} 1 & r^0 \end{array}\right] = \left[\begin{array}{cc} 0 & e_{o_N} \end{array}\right]$. Finally, note that

$$\left[\begin{array}{cc} 1 & r^0 \end{array}\right] = \left[\begin{array}{cc} 1 & \mathbf{0} \end{array}\right] + \sum_{i=1}^{N} \lfloor \rho_i \rfloor \left[\begin{array}{cc} 0 & e_i \end{array}\right]$$

Using these arguments one can show that the matrix $\left[\begin{array}{cc} e & \mathbf{R} \end{array}\right]$ can be transformed into the identity matrix of dimension $N+1$ by row operations. Therefore, $\left[\begin{array}{cc} e & \mathbf{R} \end{array}\right]$ is non-singular, i.e., the third condition of Definition 3.1 is satisfied. Moreover, the inverse of $\left[\begin{array}{cc} e & \mathbf{R} \end{array}\right]$, which will be needed during the gradient estimation part of our approach in the next section, can be evaluated to give:

$$\left[\begin{array}{cc} e & \mathbf{R} \end{array}\right]^{-1} = \left[\begin{array}{c} 1+\lfloor\rho_{o_N}\rfloor \quad \lfloor\rho_{o_{N-1}}\rfloor - \lfloor\rho_{o_N}\rfloor \quad \lfloor\rho_{o_{N-2}}\rfloor - \lfloor\rho_{o_{N-1}}\rfloor \quad \cdots \quad -\lfloor\rho_{o_1}\rfloor \\ -\hat{e}_{N+1-\bar{o}_1} + \hat{e}_{N+2-\bar{o}_1} \\ \vdots \\ -\hat{e}_{N+1-\bar{o}_N} + \hat{e}_{N+2-\bar{o}_N} \end{array}\right] \tag{18}$$

where $\hat{e}_i$ is the $(N+1)-$dimensional unit vector whose ith component is 1 and $\bar{o}$ is the N-dimensional ordering vector such that $\bar{o}_k \in \{1, ..., N\}$ satisfying the relation

$$o_i = j \Leftrightarrow \bar{o}_j = i$$

One can also verify that $\left[\begin{array}{cc} e & \mathbf{R} \end{array}\right]^{-1}$ above is such that $\left[\begin{array}{cc} e & \mathbf{R} \end{array}\right]^{-1}\left[\begin{array}{cc} e & \mathbf{R} \end{array}\right] = \mathbf{I}$. ■

We stress that the selection set $\mathcal{S}(\rho)$ is *not unique*; however, given a selection set $\mathcal{S}(\rho)$, the α_i values are unique for $i = 0, ..., N$. There are clearly different ways one can construct $\mathcal{S}(\rho)$, including randomized methods. For instance, one can start out by randomly selecting the first element of the selection set from $\mathcal{C}(\rho)$ and then proceed through a scheme similar to the one used above.

The following is an algorithmic procedure for constructing $\mathcal{S}(\rho)$ as presented in Theorem 3.1:

- Initialize the index set $I = \{1, ..., N\}$ and define a temporary vector $v = \tilde{\rho}$.
- While $I \neq \emptyset$:

1. $\tilde{r}^i = \sum_{j\in I} e_j$ where $i = \arg\min\{v_j,\ j \in I\}$
2. $\alpha_i = v_i$
3. $v \leftarrow v - \alpha_i \tilde{r}^i$
4. $I \leftarrow I \backslash \{i\}$

- $\tilde{r}^0 = 0$
- $\alpha_0 = 1 - \sum_{i=1}^{N} \alpha_i$
- $\mathcal{S}(\rho) = \{r^i | r^i = \tilde{r}^i + \lfloor\rho\rfloor \text{ for } i = 0, ..., N\}$

Example: In order to clarify our notation and illustrate the specification of the sets $\mathcal{C}(\rho)$, $\mathcal{N}(\rho)$ and $\mathcal{S}(\rho)$, we provide the following example, which we will use throughout our analysis. Consider the allocation problem of $K = 10$ resources over $N = 3$ users, and let $\rho = [3.9, 3.9, 2.2]$. The feasible set is

$$A_c = \left\{\rho : \sum_{i=1}^{N} \rho_i = 10\right\} \tag{19}$$

Since $\lfloor \rho \rfloor = [3,3,2]$, we have the unit cube

$$\mathcal{C}(\rho) = \{[3,3,2],[3,3,3],[3,4,2],[3,4,3],[4,3,2],[4,3,3],[4,4,2],[4,4,3]\}$$

and the *feasible neighbors* of ρ are

$$\mathcal{N}(\rho) = \{[3,4,3],[4,4,2],[4,3,3]\}$$

Let us now construct a selection set satisfying the conditions of Definition 3.1 using the algorithm described above. First we initialize the index set $I = \{1,2,3\}$ and the residual vector $v = \rho - \lfloor \rho \rfloor = [0.9, 0.9, 0.2]$. Note that $\arg\min_{j\in\{1,2,3\}}\{v_j\} = 3$, therefore,

$$\begin{aligned} \tilde{r}^3 &= \sum_{j\in\{1,2,3\}} e_j = [1,1,1] \\ \alpha_3 &= v_3 = 0.2 \end{aligned}$$

Next, we update

$$\begin{aligned} v &\leftarrow v - \alpha_3 \tilde{r}^3 = [0.7, 0.7, 0] \\ I &\leftarrow I\backslash\{3\} = \{1,2\} \end{aligned}$$

Now, note that the first two components in the updated v are equal. We can select any one of them for the next $\arg\min_{j\in\{1,2\}}\{v_j\}$. Thus, if we pick the first component we get

$$\begin{aligned} \tilde{r}^1 &= \sum_{j\in\{1,2\}} e_j = [1,1,0] \\ \alpha_1 &= v_1 = 0.7 \end{aligned}$$

Proceeding as before, we update

$$\begin{aligned} v &\leftarrow v - \alpha_1 \tilde{r}^1 = [0,0,0] \\ I &\leftarrow I\backslash\{1\} = \{2\} \end{aligned}$$

and finish this step by setting

$$\begin{aligned} \tilde{r}^2 &= \sum_{j\in\{2\}} e_j = [0,1,0] \\ \alpha_2 &= v_2 = 0 \end{aligned}$$

Finally,

$$\begin{aligned} v &\leftarrow v - \alpha_2 \tilde{r}^2 = [0,0,0] \\ I &\leftarrow I\backslash\{2\} = \{\} \end{aligned}$$

We may now construct the selection set from the vectors given by (16):

$$\begin{aligned} r^1 &= \tilde{r}^1 + \lfloor \rho \rfloor = [1,1,0] + [3,3,2] = [4,4,2] \\ r^2 &= \tilde{r}^2 + \lfloor \rho \rfloor = [0,1,0] + [3,3,2] = [3,4,2] \\ r^3 &= \tilde{r}^3 + \lfloor \rho \rfloor = [1,1,1] + [3,3,2] = [4,4,3] \\ r^0 &= \tilde{r}^0 + \lfloor \rho \rfloor = [0,0,0] + [3,3,2] = [3,3,2] \end{aligned}$$

i.e.,

$$\mathcal{S}(\rho) = \{[3,3,2],[3,4,2],[4,4,2],[4,4,3]\}$$

The example above illustrates the important difference between the selection set $\mathcal{N}_N(\rho)$ employed in [1] and the present construction: While all the elements of the set $\mathcal{N}_N(\rho)$ are feasible, the elements of the selection set $\mathcal{S}(\rho)$ constructed above may be infeasible. The following lemma considers resource allocation problems with total capacity constraints as a special case of discrete stochastic optimization and asserts that there is always exactly one feasible point in $\mathcal{S}(\rho)$.

Lemma 3.1 *For problems (1) with a feasible set* $A_d = \left\{ r : \sum_{i=1}^N r_i = K,\ r \in \mathbb{Z}_+^N \right\}$*, the selection set* $\mathcal{S}(\rho)$ *constructed above includes one and only one feasible point. Moreover, this point is the argument of* $\min_{r\in\mathcal{N}(\rho)} \|\rho - r\|$.

Proof. Since $\rho \in A_c$, it follows from (7) that there exist $\{\alpha_i\}_{i=1}^M$ that satisfy

$$\rho_j = \sum_{i=1}^M \alpha_i r_j^i, \quad r^i \in \mathcal{N}(\rho), \quad \sum_{i=1}^M \alpha_i = 1, \ \ \alpha_i \geq 0 \text{ for } i = 1, ..., M$$

where $M = |\mathcal{N}(\rho)|$. Therefore,

$$\sum_{j=1}^N \rho_j = \sum_{j=1}^N \sum_{i=1}^M \alpha_i r_j^i = \sum_{i=1}^M \alpha_i \sum_{j=1}^N r_j^i = \sum_{i=1}^M \alpha_i K = K$$

Then, we can write

$$\sum_{j=1}^N \lfloor \rho_j \rfloor \leq K \leq \sum_{j=1}^N \lceil \rho_j \rceil$$

where the equality only holds for integer allocations. Since we agreed that ρ does not have integer components,

$$\sum_{j=1}^N \lfloor \rho_j \rfloor < K < \sum_{j=1}^N \lceil \rho_j \rceil$$

Note that

$$\sum_{j=1}^N (\lceil \rho_j \rceil - \lfloor \rho_j \rfloor) = N \tag{20}$$

and define the residual resource capacity m, with $0 < m < N$, as

$$m = K - \sum_{j=1}^N \lfloor \rho_j \rfloor \tag{21}$$

Also note that from (12) and (16)

$$r^0 = \lfloor \rho \rfloor + \tilde{r}^0 = \lfloor \rho \rfloor \in \mathcal{S}(\rho)$$

and from (8)

$$r^{o_1} = \lfloor\rho\rfloor + \tilde{r}^{o_1} = \lfloor\rho\rfloor + \sum_{k=1}^{N} e_{o_k} = \lceil\rho\rceil \in \mathcal{S}(\rho) \tag{22}$$

Now, observe that during the construction of $\mathcal{S}(\rho)$,

$$r^{o_l} - r^{o_{l+1}} = e_{o_l}$$

therefore,

$$\sum_{i=1}^{N} r_i^{o_l} - \sum_{i=1}^{N} r_i^{o_{l+1}} = \sum_{i=1}^{N} (e_{o_l})_i = 1 \tag{23}$$

Using (23),

$$\sum_{i=1}^{N} r^{o_1} - \sum_{i=1}^{N} r^{o_{N+1-m}} = N - m$$

and it follows from (22) that

$$\sum_{i=1}^{N} r^{o_{N+1-m}} = \sum_{i=1}^{N} \lceil\rho_i\rceil - N + m = K$$

where we have used (20) and (21). Therefore, $r^{o_{N+1-m}} \in \mathcal{N}(\rho)$ and the first part of the proof is complete.

By construction of the selection set $\mathcal{S}(\rho)$, the elements r^{o_l} satisfy the following

$$r_i^{o_l} = \lceil\rho_i\rceil \text{ and } r_j^{o_l} = \lfloor\rho_j\rfloor \Rightarrow \tilde{\rho}_i \geq \tilde{\rho}_j \tag{24}$$

Therefore, we claim that $r^{o_{N+1-m}}$ is the solution of the minimization problem

$$\min_{r\in\mathcal{N}(\rho)} \|\rho - r\| = \sqrt{\sum_{i=1}^{N} (\rho_i - r_i)^2}$$

for $\rho \in A_c$. All elements $r \in \mathcal{N}(\rho)$ can be characterized in terms of a set $\mathcal{M}_r$ of indices defined as

$$\mathcal{M}_r = \{i | r_i = \lceil\rho_i\rceil\}$$

where $|\mathcal{M}_r| = m$ for $r \in \mathcal{N}(\rho)$. One can then write an equivalent minimization problem as

$$\begin{aligned} \min_{r\in\mathcal{N}(\rho)} \sum_{i=1}^{N} (\rho_i - r_i)^2 &= \min_{r\in\mathcal{N}(\rho)} \left[\sum_{\substack{i=1 \\ r_i=\lfloor\rho_i\rfloor}}^{N} (\rho_i - r_i)^2 + \sum_{\substack{i=1 \\ r_i=\lceil\rho_i\rceil}}^{N} (\rho_i - r_i)^2 \right] \\ &= \min_{r\in\mathcal{N}(\rho)} \left[\sum_{i\in I\setminus\mathcal{M}_r} \tilde{\rho}_i^{\,2} + \sum_{i\in\mathcal{M}_r} (1 - \tilde{\rho}_i)^2 \right] \end{aligned}$$

For $r, r^{o_{N+1-m}} \in \mathcal{N}(\rho)$, the sets $\mathcal{M}_r$ and $\mathcal{M}_{r^{o_{N+1-m}}}$ are formed by m elements from the set $\{1, ..., N\}$. Starting at $\mathcal{M}_{r^{o_{N+1-m}}}$, one can reach any $\mathcal{M}_r$ by a series of iterations, each iteration involving the removal of one element from the set $\mathcal{M}_{r^{o_{N+1-m}}} \backslash \mathcal{M}_r$, and the addition of an element from the set $\mathcal{M}_r \backslash \mathcal{M}_{r^{o_{N+1-m}}}$. If we remove i and add j while $\tilde{\rho}_i \geq \tilde{\rho}_j$ we increase the objective value by

$$\tilde{\rho}_i^2 - \tilde{\rho}_j^2 + (1 - \tilde{\rho}_j)^2 - (1 - \tilde{\rho}_i)^2 = 2(\tilde{\rho}_i - \tilde{\rho}_j) \geq 0$$

Since $\mathcal{M}_{r^{o_{N+1-m}}}$ has the arguments for the m highest $\tilde{\rho}_i$, we cannot decrease the distance by moving to another $r \in \mathcal{N}(\rho)$ therefore $r^{o_{N+1-m}}$ minimizes the distance from ρ. ■

Example (Cont.): For the resource allocation problem of $K = 10$ resources over $N = 3$ users, given $\rho = [3.9, 3.9, 2.2]$, we found a selection set

$$\mathcal{S}(\rho) = \{[3, 3, 2], [3, 4, 2], [4, 4, 2], [4, 4, 3]\}$$

Observe that $[4, 4, 2]$ is the only element of $\mathcal{S}(\rho)$ above which is feasible, consistent with Lemma 3.1. In addition, note that $[4, 4, 2]$ is the obvious solution of the minimization problem

$$\min_{r \in \mathcal{N}(\rho)} \|\rho - r\|$$

where $\mathcal{N}(\rho) = \{[3, 4, 3], [4, 4, 2], [4, 3, 3]\}$.

4 Construction of surrogate cost functions and their gradients

Since our approach is based on iterating over the continuous state $\rho \in A_c$, yet drive the iteration process with information involving $L_d(r)$ obtained from a sample path under r, we must establish a relationship between $L_d(r)$ and $\bar{L}_c(\rho)$. The choice of $\bar{L}_c(\rho)$ is rather flexible and may depend on information pertaining to a specific model and the nature of the given cost $L_d(r)$.

Before defining $\bar{L}_c(\rho)$, we shall concentrate on surrogate cost functions $L_c(\rho, \mathcal{S}(\rho), \omega)$ (which clearly depend on a selection set and a sample path) that satisfy the following two conditions:

(C1): *Consistency*: $L_c(r, \mathcal{S}(r), \omega) = L_d(r, \omega)$ for all $r \in \mathbb{Z}_+^N$.

(C2): *Piecewise Linearity:* $L_c(\rho, \mathcal{S}(\rho), \omega)$ is a linear function of ρ over *conv*$(\mathcal{S}(\rho))$.

In the sequel, the '$\mathcal{S}(\rho)$' term will be dropped along with 'ω' for simplicity.

Consistency is an obvious requirement for $L_c(\rho)$. Piecewise linearity is chosen for convenience, since manipulating linear functions over *conv*$(\mathcal{S}(\rho))$ simplifies analysis, as will become clear in what follows.

Given some state $\rho \in A_c$ and cost functions $L_d(r^j)$ for all $r^j \in \mathcal{S}(\rho)$, it follows from **(C2)** and (17) that we can write

$$\rho = \sum_{j=0}^{N} \alpha_j r^j \Rightarrow L_c(\rho) = \sum_{j=0}^{N} \alpha_j L_c(r^j)$$

with $\sum_{j=0}^{N} \alpha_j = 1$, $\alpha_j \geq 0$ for all $j = 0, .., N$. Moreover, by **(C1)**, we have

$$L_c(\rho) = \sum_{j=0}^{N} \alpha_j L_d(r^j) \tag{25}$$

that is, $L_c(\rho)$ is a convex combination of the costs of discrete neighbors in $\mathcal{S}(\rho)$. Note that although $\mathcal{S}(\rho)$ is not unique, given $\mathcal{S}(\rho)$, the values of α_i for $i = 0, ..., N$ are unique; therefore, $L_c(\rho)$ is well defined.

We now define a surrogate cost function $\bar{L}_c(\rho)$ and the corresponding selection set $\mathcal{S}^*(\rho)$ as

$$\bar{L}_c(\rho) = \min_{\mathcal{S}(\rho)} L_c(\rho) \tag{26}$$

and

$$\mathcal{S}^*(\rho) = \arg\min_{\mathcal{S}(\rho)} L_c(\rho) \tag{27}$$

where the minimization is over all possible selection sets for the point ρ. The function $\bar{L}_c(\rho)$ satisfies the consistency condition **(C1),** but it may not be a continuous function due to changes in the selection set $\mathcal{S}^*(\cdot)$ for neighboring points.

Next, if we are to successfully use the iterative scheme described by (4)-(5), we need information of the form $H_n(\rho_n, r_n, \omega_n)$ following the nth step of the on-line optimization process. Typically, this information is contained in an estimate of the gradient. Our next objective, therefore, is to seek the selection-set-dependent sample gradient $\nabla L_c(\rho)$ expressed in terms of directly observable sample path data.

4.1 Gradient evaluation

The gradient information necessary to drive the stochastic approximation part of the surrogate method is evaluated depending on the form of the cost function. Gradient estimation for separable cost functions is significantly simpler and is discussed in Section 4.3.

Since $L_c(\rho)$ is a linear function on the convex hull defined by $\mathcal{S}(\rho)$, one can write

$$L_c(\rho) = \sum_{i=1}^{N} \beta_i \rho_i + \beta_0 \tag{28}$$

where

$$\beta_i = \frac{\partial L_c(\rho)}{\partial \rho_i}, \; i = 1, ..., N$$

and $\beta_0 \in \mathbb{R}$ is a constant. Note that the β_i values depend on the selection set $\mathcal{S}(\rho)$, which, as already pointed out, may not be unique.

For any $r^j \in \mathcal{S}(\rho)$, one can use (28) and **(C1)** to write

$$L_d(r^j) = \sum_{i=1}^{N} \beta_i r_i^j + \beta_0$$

Note that the values $L_d(r^j)$ are either observable or can be evaluated using Concurrent Estimation or Perturbation Analysis techniques (see [15], [16]) despite the fact that $r^j \in \mathcal{S}(\rho)$ may be infeasible, i.e., *having infeasible points in the selection set does not affect our ability to obtain gradients.* One can now rewrite the equation above as

$$\left[\begin{array}{cc} e & \mathbf{R} \end{array} \right] \beta = L$$

where e is an $(N+1)-$dimensional column vector with all entries equal to 1, $\beta = [\beta_0, ..., \beta_N]'$, $\mathbf{R}$ is the matrix whose rows are $r^j \in \mathcal{S}(\rho)$, and L is the column vector of costs for these discrete states. Since we have constructed $\mathcal{S}(\rho)$ so that $\left[\begin{array}{cc} e & \mathbf{R} \end{array} \right]$ is non-singular, the gradient given by $\nabla L_c(\rho) = [\beta_1, ..., \beta_N]'$, can be obtained from the last equation as

$$\nabla L_c(\rho) = \left[\begin{array}{cc} \mathbf{0} & \mathbf{I} \end{array} \right] \left[\begin{array}{cc} e & \mathbf{R} \end{array} \right]^{-1} L \tag{29}$$

where $\mathbf{I}$ is the identity matrix of dimension N and $\mathbf{0}$ is the N-dimensional vector of zeros. Substituting from equation (18), the gradient can be written as

$$\nabla L_c(\rho) = \left[\begin{array}{c} -\hat{e}_{N+1-\bar{o}_1} + \hat{e}_{N+2-\bar{o}_1} \\ \vdots \\ -\hat{e}_{N+1-\bar{o}_N} + \hat{e}_{N+2-\bar{o}_N} \end{array} \right] \left[\begin{array}{c} L_d(r^0) \\ L_d(r^{o_N}) \\ \vdots \\ L_d(r^{o_1}) \end{array} \right] \tag{30}$$

Therefore,

$$\nabla_j L_c(\rho) = L_d(r^j) - L_d(r^k) \tag{31}$$

where k satisfies $\bar{o}_j + 1 = \bar{o}_k$, i.e., $r^j - r^k = e_j$. As pointed out in the Introduction, this is a significant simplification over the gradient evaluation used in our earlier work [1]. Moreover, this analysis allows us to combine the algorithm for determining the selection set given in the previous section with the gradient estimation component of our approach to obtain the following:

- Initialize the index set $I = \{1, ..., N\}$
- $r^i = \lceil \rho \rceil$ where $i = \arg\min_{j \in I} \tilde{\rho}_j$
- $OC = L_d(r^i)$
- $oi = i$
- $I = I \backslash \{i\}$
- While $I \neq \emptyset$:

1. $r^k = r^{oi} - e_{oi}$ where $k = \arg\min_{j \in I} \tilde{\rho}_j$
2. $\nabla_{oi} L_c(\rho) = OC - L_d(r^k)$
3. $OC = L_d(r^k)$
4. $oi = k$

5. $I = I \backslash \{k\}$

- $r^0 = \lfloor \rho \rfloor$
- $\nabla_{oi} L_c(\rho) = OC - L_d(r^0)$

Example (Cont.): For the example we have been using throughout the paper, consider the cost function

$$J(r) = \|r - [2,5,3]\|^2$$

Let $\rho_n = [3.9, 3.9, 2.2]$, for which we found $\mathcal{S}(\rho) = \{[3,3,2],[3,4,2],[4,4,2],[4,4,3]\}$ with $r^1 = [4,4,2]$, $r^2 = [3,4,2]$, $r^3 = [4,4,3]$, and $r^0 = [3,3,2]$. The gradient at this point can, therefore, be evaluated using (31) to give

$$\nabla L_c(\rho_n) = \left[\begin{array}{c} J_d([4,4,2]) - J_d([3,4,2]) \\ J_d([3,4,2]) - J_d([3,3,2]) \\ J_d([4,4,3]) - J_d([4,4,2]) \end{array} \right] = \left[\begin{array}{c} 3 \\ -3 \\ -1 \end{array} \right]$$

Using $\eta_n = 0.5$ in (4):

$$\begin{aligned} \rho_{n+1} &= \pi_{n+1}[\rho_n - \eta_n \nabla L_c(\rho_n)] \\ &= \pi_{n+1}[[2.4, 5.4, 2.7]] \\ &= [2.2, 5.2, 2.6] \end{aligned}$$

where we have used the projection π to map the point $[2.4, 5.4, 2.7]$ into a feasible point $[2.2, 5.2, 2.6] \in A_c$. For this example, π can be defined as follows:

$$\pi[\bar{\rho}] = \arg \min_{\sum_{i=1}^{N} \rho_i = 10} \|\rho - \bar{\rho}\|$$

4.2 Projection Mapping

The projection mapping π is a crucial element of our method and may have a significant effect on convergence. In this section, we discuss a projection mapping which can be used for resource allocation problems with feasible sets

$$A_d = \left\{ r : \sum_{i=1}^{N} r_i = K,\ r \in \mathbb{Z}_+^N \right\}$$

Consider the optimization problem

$$\min_{\rho} \sum_{i=1}^{N} (\rho_i - \bar{\rho}_i)^2$$

subject to

$$\begin{aligned} \rho_i &\geq 0 \\ \sum_{i=1}^{N} \rho_i &= K \end{aligned}$$

The solution to this optimization problem, which we will denote by $\pi(\bar{\rho})$, is the closest point in the feasible set A_c to the point $\bar{\rho}$. Note that a π projection to a closed convex set defined in this manner is continuous and nonexpansive, therefore it guarantees convergence (see [1]).

Let us consider the relaxed problem

$$\min_{\rho_i \geq 0} \sum_{i=1}^{N} \left[(\rho_i - \bar{\rho}_i)^2 - \lambda \rho_i\right] + \lambda K$$

The necessary optimality conditions are

$$\begin{aligned} [2(\rho_i - \bar{\rho}_i) - \lambda] &= 0 \text{ for } \rho_i > 0 \\ [2(\rho_i - \bar{\rho}_i) - \lambda] &> 0 \text{ for } \rho_i = 0 \\ \sum_{i=1}^{N} \rho_i &= K \end{aligned}$$

or equivalently

$$\begin{aligned} \rho_i &= \bar{\rho}_i + \frac{\lambda}{2} \text{ for } \rho_i > 0 \\ \rho_i &> \bar{\rho}_i + \frac{\lambda}{2} \text{ for } \rho_i = 0 \\ \sum_{i=1}^{N} \rho_i &= K \end{aligned}$$

i.e.,

$$\begin{aligned} \rho_i &= \max(0, \bar{\rho}_i + \frac{\lambda}{2}) \\ \sum_{i=1}^{N} \rho_i &= K \end{aligned}$$

These equations suggest the following algorithm for the π projection:

Projection Algorithm:

- Initialize $\lambda^0 = \frac{2}{N}(K - \sum_{i=1}^{N} \max(0, \bar{\rho}_i))$
- If some stopping condition is not satisfied:

1. For $i = 1, 2, ...N$, $\qquad \rho_i = \max(0, \bar{\rho}_i + \frac{\lambda}{2})$
2. $\lambda \leftarrow \lambda + \frac{2}{N}(K - \sum_{i=1}^{N} \rho_i)$

- Set $\pi[\bar{\rho}] = \rho$.

A common stopping condition we have used in our work (see also Section 7) is $\left| K - \sum_{i=1}^{N} \rho_i \right| \leq \varepsilon$, for some small $\varepsilon > 0$. Then, the vector ρ is rescaled

$$\rho \leftarrow \frac{K}{\sum_{i=1}^{N} \rho_i} \rho$$

to satisfy the capacity constraint. The error introduced while rescaling is small and it is a monotonically increasing function of ε. Note that there is a trade-off between the number of iterations needed and the size of the resulting error term determined by the selection of ε.

4.3 Separable cost functions

Suppose that the cost function, $L_d(\cdot)$, is *separable* in the sense that it is a sum of component costs each dependent on its local state only, i.e., let

$$L_d(r) = \sum_{i=1}^{N} L_{d,i}(r_i) \tag{32}$$

In this case, our approach is significantly simplified. In particular, from (31) and (32), we can write

$$\begin{aligned}
\nabla_j L_c(\rho) &= L_d(r^j) - L_d(r^k) \\
&= \sum_{i=1}^{N} L_{d,i}(r_i^j) - \sum_{i=1}^{N} L_{d,i}(r_i^k) \\
&= \sum_{i=1}^{N} [L_{d,i}(r_i^j) - L_{d,i}(r_i^k)] \\
&= L_{d,j}(r_j^j) - L_{d,j}(r_j^j - 1)
\end{aligned}$$

where k satisfies $\bar{o}_j + 1 = \bar{o}_k$, i.e., $r^j - r^k = e_j$. Note that $r_j^j = \lceil \rho_j \rceil$ and $r_j^j - 1 = \lfloor \rho_j \rfloor$; therefore,

$$\nabla_j L_c(\rho) = L_{d,j}(\lceil \rho_j \rceil) - L_{d,j}(\lfloor \rho_j \rfloor) \tag{33}$$

This result indicates that for separable cost functions estimating sensitivities does not require the determination of a selection set; we can instead simply pick a feasible neighbor (preferably the closest feasible neighbor to ρ) and apply Perturbation Analysis (PA) techniques to determine the gradient components through (33). There are a number of PA techniques developed precisely for evaluating the effect of decreasing and increasing the number of resources allocated to user

i; for example, estimating the sensitivity of packet loss in a radio network with respect to adding/removing a transmission slot available to the ith user [17], [18]. In [19] a PA technique is used together with our methodology to solve a call admission problem (with a separable cost function) over a communication network where there are capacity constraints on each node, while there is no total capacity constraint for the network.

5 Recovery of optimal discrete states

Our ultimate goal remains the solution of (1), that is the determination of $r^* \in A_d$ that solves this optimization problem. Our approach is to solve (2) by iterating on $\rho \in A_c$ and, at each step, transforming ρ through some $f \in \mathcal{F}_\rho$. The connection between ρ and $r = f(\rho)$ for each step is therefore crucial, as is the relationship between ρ^* and $f(\rho^*)$ when and if this iterative process comes to an end identifying a solution ρ^* to the surrogate problem (2).

The following theorem identifies a key property of the selection set $\mathcal{S}^*(\rho^*)$ of an *optimal* surrogate state ρ^*.

Theorem 5.1 *Let ρ^* minimize $\bar{L}_c(\rho)$ over A_c. If $r^* = \arg\min_{r\in\mathcal{S}^*(\rho^*)} L_d(r) \in \mathcal{N}(\rho^*)$, i.e., the minimal cost element r^* of the selection set $\mathcal{S}^*(\rho^*)$ corresponding to $\bar{L}_c(\rho^*)$ is feasible, then r^* minimizes $L_d(r)$ over A_d and satisfies $L_d(r^*) = \bar{L}_c(\rho^*)$.*

Proof. By (25), the optimal surrogate state $\rho^* = \arg\min_{\rho\in A_c} \bar{L}_c(\rho)$ satisfies

$$\bar{L}_c(\rho^*) = \sum_{j=0}^{N} \alpha_j L_d(r^j)$$

where $\sum_{j=0}^{N} \alpha_j = 1$, $r^j \in \mathcal{S}^*(\rho^*)$, $\alpha_j \geq 0$ for $j = 0, .., N$. Then,

$$\bar{L}_c(\rho^*) = \sum_{j=0}^{N} \alpha_j L_d(r^j) \geq \sum_{j=0}^{N} \alpha_j L_d(r^*) = L_d(r^*) \tag{34}$$

regardless of the feasibility of r^*.

Next, note that $A_d \subset A_c$ and $\bar{L}_c(r) = L_d(r)$ for any $r \in A_d$. Therefore, if $r^* \in \mathcal{N}(\rho^*)$, then

$$L_d(r^*) = \bar{L}_c(r^*) \geq \bar{L}_c(\rho^*) \tag{35}$$

In view of (34) and (35), we then get

$$L_d(r^*) \leq \bar{L}_c(\rho^*) \leq L_d(r^*)$$

It follows that

$$L_d(r^*) = \bar{L}_c(\rho^*)$$

that is, r^* is optimal over A_d. Finally, since r^* is one of the discrete feasible neighboring states of ρ^*, i.e. $r^* \in \mathcal{N}(\rho^*)$, we have $r^* = f(\rho^*)$ for some $f \in \mathcal{F}_{\rho^*}$. ■

Corollary 5.1 *For unconstrained problems, let ρ^* minimize $\bar{L}_c(\rho)$. Then,*

$$r^* = \arg \min_{r \in \mathcal{S}^*(\rho^*)} L_d(r)$$

minimizes $L_d(r)$ and satisfies $L_d(r^*) = \bar{L}_c(\rho^*)$.

Proof. If problem (1) is unconstrained, then trivially $r^* = \arg\min_{r \in \mathcal{S}^*(\rho^*)} L_d(r) \in \mathcal{N}(\rho^*)$ and the result follows. ■

An interesting example of an unconstrained problem is that of lot sizing in manufacturing systems (see [20]) where the sizes of lots of different parts being produced may take any (non-negative) integer value. Clearly, Corollary 5.1 also holds for problems where the optimal point is in the interior of the feasible set where the constraints are not active, i.e., $\mathcal{N}(\rho^*) = \mathcal{C}(\rho^*)$.

If there are active constraints around the optimal point ρ^*, i.e., $\mathcal{N}(\rho^*) \neq \mathcal{C}(\rho^*)$, and there are infeasible points in the selection set $\mathcal{S}(\rho^*)$, then, if one of these infeasible points has the minimal cost, the recovery of the optimal as a feasible neighbor of ρ^* becomes difficult to guarantee theoretically, even though empirical evidence shows that this is indeed the case. This is the price to pay for the generalization of the surrogate problem method we have presented here through the introduction of a selection set that allows the inclusion of infeasible points. However, if the cost function $L_d(r)$ is "smooth" in some sense, the minimal cost element of $\mathcal{N}(\rho^*)$ will in general be either an optimal or a near-optimal point as stated in the next lemma.

Lemma 5.1 *If the cost function $L_d(r)$ satisfies*

$$\left|L_d(r^1) - L_d(r^2)\right| \leq c_\omega \left\|r^1 - r^2\right\|, \quad c_\omega < \infty \tag{36}$$

then all $r \in \mathcal{N}(\rho^*)$ satisfy

$$L_d(r) \leq \bar{L}_c(\rho^*) + c_\omega\sqrt{N} \tag{37}$$

Proof. Note that $\mathcal{S}^*(\rho^*) \subset \mathcal{C}(\rho^*)$ and $\mathcal{N}(\rho^*) \subset \mathcal{C}(\rho^*)$. It is easy to show that for $r^1, r^2 \in \mathcal{C}(\rho^*)$

$$\left\|r^1 - r^2\right\| \leq \sqrt{N}$$

By (34), there exists $r^* \in \mathcal{S}^*(\rho^*)$ that satisfies $\bar{L}_c(\rho^*) \geq L_d(r^*)$ regardless of its feasibility. For $r \in \mathcal{N}(\rho^*)$, we can write $\bar{L}_c(\rho^*) \leq L_d(r)$, therefore

$$|L_d(r) - L_d(r^*)| = L_d(r) - L_d(r^*) \geq L_d(r) - \bar{L}_c(\rho^*) \tag{38}$$

By assumption (36),

$$|L_d(r) - L_d(r^*)| \leq c_\omega \left\|r - r^*\right\| \leq c_\omega\sqrt{N} \tag{39}$$

Hence, from (38) and (39),

$$L_d(r) \leq \bar{L}_c(\rho^*) + c_\omega\sqrt{N}$$

■

In practice, for many cost metrics such as throughput or mean system time in queueing models, it is common to have the costs in the neighborhood of an optimal point be relatively close, in which case the value of c_ω is small and (37) is a useful bound.

6 Optimization Algorithm

Summarizing the results of the previous sections and combining them with the basic scheme described by (4)-(5), we obtain the following optimization algorithm for the solution of the basic problem in (1):

- Initialize $\rho_0 = r_0$ and perturb ρ_0 to have all components non-integer.
- For any iteration $n = 0, 1, \ldots$:
  1. Determine $\mathcal{S}(\rho_n)$ [using the construction of Theorem 3.1; recall that this set is generally not unique].
  2. Select $f_n \in \mathcal{F}_{\rho_n}$ such that $r_n = \arg\min_{r \in \mathcal{N}(\rho_n)} \|r - \rho_n\| = f_n(\rho_n) \in \mathcal{N}(\rho_n)$.
  3. Operate at r_n to collect $L_d(r^i)$ for all $r^i \in \mathcal{S}(\rho_n)$ [using Concurrent Estimation or some form of Perturbation Analysis; or, if feasible, through off-line simulation].
  4. Evaluate $\nabla L_c(\rho_n)$ [using (31)].
  5. Update the continuous state: $\rho_{n+1} = \pi_{n+1}[\rho_n - \eta_n \nabla L_c(\rho_n)]$.
  6. If some stopping condition is not satisfied, repeat steps for $n+1$. Else, set $\rho^* = \rho_{n+1}$.
- Obtain the optimal (or the near optimal) state as one of the neighboring feasible states in the set $\mathcal{N}(\rho^*)$.

Note that for separable cost functions, steps 1-6 can be replaced by

1. Select f_n such that $r_n = \arg\min_{r \in \mathcal{N}(\rho_n)} \|r - \rho_n\| = f_n(\rho_n) \in \mathcal{N}(\rho_n)$.
2. Operate at r_n to evaluate $\nabla L_c(\rho_n)$ using Perturbation Analysis and (33).
3. Update the continuous state: $\rho_{n+1} = \pi_{n+1}[\rho_n - \eta_n \nabla L_c(\rho_n)]$.
4. If some stopping condition is not satisfied, repeat steps for $n+1$. Else, set $\rho^* = \rho_{n+1}$.

The surrogate part of this algorithm is a stochastic approximation scheme with projection whose convergence was analyzed in [1] and references therein.

Note that ideally we would like to have $\nabla J_c(\rho_n)$ be the cost sensitivity driving the algorithm. Since this information is not always available in a stochastic environment and since $J_c(\rho_n) = E[\bar{L}_c(\rho_n, \omega)]$, the stochastic approximation algorithm uses $\nabla \bar{L}_c(\rho_n, \omega)$ as an estimate and under some standard assumptions on the estimation error ε_n where

$$\nabla J_c(\rho_n) = \nabla \bar{L}_c(\rho_n, \omega) + \varepsilon_n$$

the convergence is guaranteed. In order to get $\nabla \bar{L}_c(\rho_n, \omega)$, however, one needs to consider all possible selection sets. In this algorithm we utilize only one of those selection sets and approximate $\nabla \bar{L}_c(\rho_n, \omega)$ with $\nabla L_c(\rho_n, S(\rho_n), \omega)$. This approximation introduces yet another error term $\bar{\varepsilon}_n$ where

$$\nabla \bar{L}_c(\rho_n, \omega) = \nabla L_c(\rho_n, S(\rho_n), \omega) + \bar{\varepsilon}_n$$

Note that this error term $\bar{\varepsilon}_n$ exists regardless of stochasticity, unless the cost function $L_d(.)$ is separable (all selection sets will yield the same sensitivity for separable cost functions). We can combine error terms to define $\epsilon_n = \bar{\varepsilon}_n + \varepsilon_n$ and write

$$\nabla J_c(\rho_n) = \nabla L_c(\rho_n, S(\rho_n), \omega) + \epsilon_n$$

If the augmented error term ϵ_n satisfies the standard assumptions, then convergence of the algorithm to the optimal follows in the same way as presented in [1].

7 Numerical Examples and Applications

We first illustrate our approach by means of a simple deterministic example, followed by a more challenging stochastic optimization application for a classical problem in manufacturing systems.

Example 1: Consider an allocation problem of $K = 20$ resources over $N = 4$ users so as to minimize the convex cost function $J_d(r)$ defined as

$$J_d(r) = \|r - [4, 5, 3, 8]\|^2$$

Suppose the initial state is $\rho_0 = [1.8, 9.1, 6.2, 2.9]$. Note that the set of feasible neighboring states $\mathcal{N}(\rho_0)$ is

$$\mathcal{N}(\rho_0) = \{[2, 10, 6, 2], [2, 9, 7, 2], [2, 9, 6, 3], [1, 10, 7, 2], [1, 10, 6, 3], [1, 9, 7, 3]\}$$

Following the steps shown in the algorithm of Section 6, we have:

1. Determine the selection set $\mathcal{S}(\rho_0)$

$$\mathcal{S}(\rho_0) = \{[1, 9, 6, 2], [1, 9, 6, 3], [2, 9, 6, 3], [2, 9, 7, 3], [2, 10, 7, 3]\}$$

2. Select $r_0 = f_0(\rho_0) \in \mathcal{N}(\rho_0)$:

$$r_0 = [2, 9, 6, 3]$$

3. Evaluate cost functions for states in $\mathcal{S}(\rho_0)$:

$$\begin{array}{lll} J_d([1,9,6,2]) = 70 & J_d([1,9,6,3]) = 59 & J_d([2,9,6,3]) = 54 \\ J_d([2,9,7,3]) = 61 & J_d([2,10,7,3]) = 70 & \end{array}$$

4. Evaluate the gradient of the cost at ρ_0

$$\begin{aligned} (\nabla J_c(\rho_0))_1 &= J_d([2,9,6,3]) - J_d([1,9,6,3]) = -5 \\ (\nabla J_c(\rho_0))_2 &= J_d([2,10,7,3]) - J_d([2,9,7,3]) = 9 \\ (\nabla J_c(\rho_0))_3 &= J_d([2,9,7,3]) - J_d([2,9,6,3]) = 7 \\ (\nabla J_c(\rho_0))_4 &= J_d([1,9,6,3]) - J_d([1,9,6,2]) = -11 \end{aligned}$$

Therefore,

$$\nabla J_c(\rho_0) = \begin{bmatrix} -5 \\ 9 \\ 7 \\ -11 \end{bmatrix}$$

5. Update the surrogate state:

$$\rho_1 = \pi_1[\rho_0 - \eta_0 \nabla J_c(\rho_0)]$$

6. If the stopping condition is not satisfied, go to step 1 and repeat with ρ_{n+1} replacing ρ_n for $n = 0, 1, ...$

Using a step size sequence $\eta_n = 0.5/(n+1)$, the following table shows the evolution of the algorithm for the first few steps. Note that the optimal allocation $[4, 5, 3, 8]$ is reached after a single step.

STEP	ρ	r	$J_c(\rho)$	$J(r)$
0	$[1.800, 9.100, 6.200, 2.900]$	$[2, 9, 6, 3]$	56.84	54
1	$[4.300, 4.600, 2.700, 8.400]$	$[4, 5, 3, 8]$	0.50	0
2	$[4.050, 4.850, 2.950, 8.150]$	$[4, 5, 3, 8]$	0.05	0

Table 1: Optimal Resource Allocation

Example 2: Consider a manufacturing system formed by five stages in series. The arrival process to the system is Poisson with rate $\lambda = 1.0$ and the service processes are all exponential with rates $\mu_1 = 2.0$, $\mu_2 = 1.5$, $\mu_3 = 1.3$, $\mu_4 = 1.2$, and $\mu_5 = 1.1$. Note that Poisson arrival process and exponential service times are not required by the algorithm. They are chosen for simplicity of the simulations.

We would like to allocate kanban (tickets) to stages $2-5$, to minimize a cost function that has two components

$$J(r) = J_1(r) + J_2(r)$$

where $r \in \mathbb{Z}_+^4$ is the vector of kanban allocated to stages $2-5$. The first component $J_1(r)$ is the average system time for jobs and the second component $J_2(r)$ is a cost on the number of kanban allocated defined as

$$J_2(r) = c\left|K - \sum_{i=1}^{4} r_i\right|$$

For large enough c, the second component $J_2(r)$ dominates the cost; therefore, a capacity constraint of the form

$$\sum_{i=1}^{4} r_i = K$$

is enforced. The problem, then, can be written as

$$\min_{\sum_{i=1}^{4} r_i = K} J_1(r)$$

which was considered in [21] with $K = 13$. The surrogate method for the same problem performs as follows:

Iterations	r	$J(r)$
1	$[3,3,3,4]$	0.798133
2	$[1,2,2,8]$	0.781896
3	$[1,5,4,8]$	0.767171
4	$[1,4,6,7]$	0.746568
5	$[1,4,6,7]$	0.761161
6	$[1,4,6,6]$	0.709394
7	$[1,3,5,6]$	0.827928
8	$[1,3,5,6]$	0.788815
9	$[1,3,5,5]$	0.730709
10	$[1,3,5,6]$	0.742748
11	$[1,3,5,5]$	0.791522
12	$[1,2,5,5]$	0.865436
13	$[1,3,4,5]$	0.795680
14	$[1,3,4,5]$	0.738700
15	$[1,3,4,5]$	0.857133
16	$[1,3,4,5]$	0.679464
17	$[1,3,4,5]$	0.875472
18	$[1,3,4,5]$	0.840447

Table 2: Optimal Kanban Allocation

At each iteration we observe 100 departures and use the decreasing step size $\eta_n = \frac{140}{n}$. The optimal allocation is observed as $[1,3,4,5]$ which matches the result from [21]. It is worthwhile noting that this optimal point is identified within 13 iterations, illustrating the convergence speed of this method.

7.1 Multicommodity Resource Allocation Problems

An interesting class of discrete optimization problems arises when Q different types of resources must be allocated to N users. The corresponding optimization problem we would like to solve is

$$\min_{r \in A_d} J(r)$$

where $r = [r_{1,1}, \dots, r_{1,Q}, \cdots, r_{N,1}, \dots, r_{N,Q}]$ is the allocation vector and $r_{i,q}$ is the number of resources of type q allocated to user i. A typical feasible set A_d is defined by the capacity constraints

$$\sum_{i=1}^{N} r_{i,q} \leq K_q, \quad q = 1, \dots, Q$$

and possibly additional constraints such as $\beta_i \leq r_{i,q} \leq \gamma_i$ for $i = 1, \dots, N$. Aside from the fact that such problems are of higher dimensionality because of the Q different resource types that must be allocated to each user, it is also common that they exhibit multiple local minima. Examples of such problems are encountered in operations planning that involve N tasks to be simultaneously performed, each task i requiring a "package" of resources $(r_{i,1}, \ldots, r_{i,Q})$ in order to be carried out. The natural trade-off involved is between carrying out fewer tasks each with a high probability of success (because each task is provided adequate resources) and carrying out more tasks each with lower probability of success.

The "surrogate problem" method provides an attractive means of dealing with these problems with local minima because of its convergence speed. Our approach for solving these problems is to randomize over the initial states r_0 (equivalently, ρ_0) and seek a (possibly local) minimum corresponding to this initial point. The process is repeated for different, randomly selected, initial states so as to seek better solutions. For deterministic problems, the best allocation seen so far is reported as the optimal. For stochastic problems, we adopt the stochastic comparison approach in [10]. The algorithm is run from a randomly selected initial point and the cost of the corresponding final point is compared with the cost of the "best point seen so far". The stochastic comparison test in [10] is applied to determine the "best point seen so far" for the next run. Therefore, the surrogate problem method can be seen as a complementary component for random search algorithms that exploits the problem structure to yield better generating probabilities (as discussed in [10]), which will eliminate (or decrease) the visits to poor allocations enabling them to be applied on-line.

In what follows we consider a problem with $N = 16$, $Q = 2$, and $K_1 = 20$, $K_2 = 8$. We then seek a 32−dimensional vector $r = [r_{1,1}, r_{1,2}, \cdots, r_{16,1}, \ldots, r_{16,2}]$ to maximize a reward function of the form

$$J(r) = \sum_{i=1}^{16} J_i(r) \tag{40}$$

subject to

$$\sum_{i=1}^{N} r_{i,1} \leq 20, \qquad \sum_{i=1}^{N} r_{i,2} \leq 8$$

The reward functions $J_i(r)$ we will use in this problem are defined as

$$J_i = V_i P_i^0(r) - C_1 r_{i,1} P_i^1(r) - C_2 r_{i,2} P_i^2(r) \tag{41}$$

In (41), V_i represents the "value" of successfully completing the ith task and $P_i^0(r)$ is the probability of successful completion of the ith task under an allocation r. In addition, C_q is the cost of a resource of type q, where $q = 1, 2$, and $P_i^q(r)$ is the probability that a resource of type q is completely consumed or lost during the execution of the ith task under an allocation r. A representative example of a reward function for a single task with $V_i = 150$ is shown in Fig. 1. The cost values of resource types are $C_1 = 20$ and $C_2 = 40$, and the values for tasks we will use in this problem range between 50 and 150.

The surrogate method is executed from random initial points and the results for some runs are shown in Fig. 2. Note that due to local maxima, some runs yield suboptimal results. However, in all cases convergence is attained extremely fast, enabling us to repeat the optimization process multiple times with different initial points in search of the global maximum. Although it is infeasible to identify the actual global maximum, we have compared our approach to a

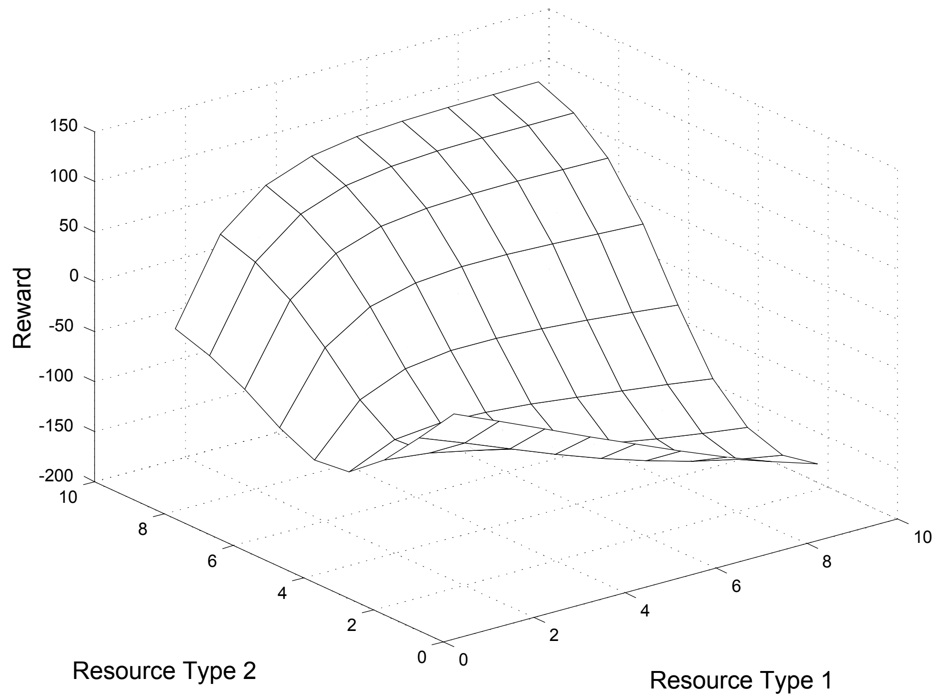

Figure 1: A typical reward function $J_i(r_{i,1}, r_{i,2})$

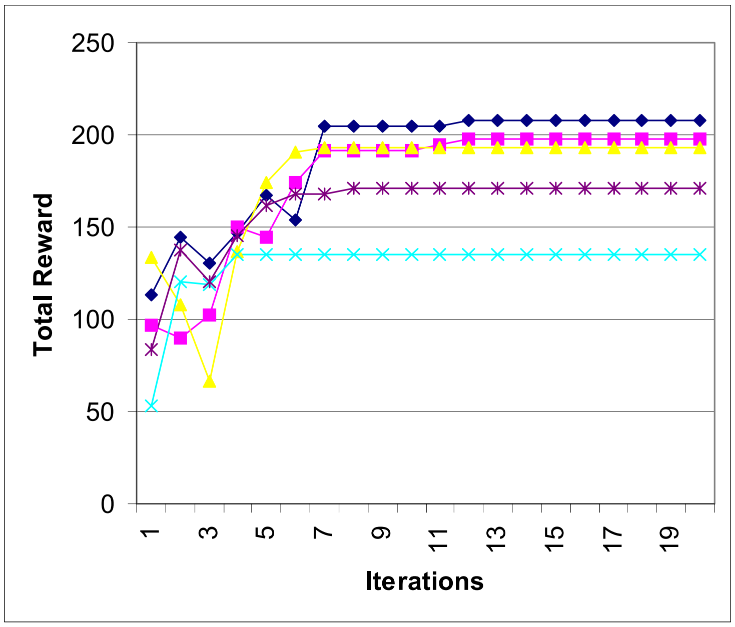

Figure 2: Algorithm convergence under different initial points

few heuristic techniques and pure random search methods and found the "surrogate problem" method to outperform them.

8 Conclusions

In this paper we have generalized the methodology presented in [1] for solving stochastic discrete optimization problems. In particular, we have introduced the concept of a "selection set" associated with every surrogate state $\rho \in A_c$ and modified the definition of the surrogate cost function $\bar{L}_c(\rho)$ so that the method can be applied to arbitrary constraint sets and is computationally more efficient.

As in [1], the discrete optimization problem was transformed into a "surrogate" continuous optimization problem which was solved using gradient-based techniques. It was shown that, under certain conditions, the solution of the original problem is recovered from the optimal surrogate state. A key contribution of the methodology is its *on-line* control nature, based on the actual data from the underlying system. One can therefore see that this approach is intended to combine the advantages of a stochastic approximation type of algorithm with the ability to obtain sensitivity estimates with respect to discrete decision variables. This combination leads to very fast convergence to the optimal point.

Using this approach, we have also tackled a class of particularly hard multicommodity discrete optimization problems, where multiple local optima typically exist. Exploiting the convergence speed of the surrogate method, we presented a procedure where the algorithm is started from multiple random initial states in an effort to determine the global optimum.

References

[1] K. Gokbayrak and C. G. Cassandras, "An on-line 'surrogate problem' methodology for stochastic discrete resource allocation problems," *J. of Optimization Theory and Applications*, 2000. To appear.

[2] C. G. Cassandras and C. G. Panayiotou, "Concurrent sample path analysis of discrete event systems," *Journal of Discrete Event Dynamic Systems: Theory and Applications*, vol. 9, pp. 171–195, 1999.

[3] H. Hafner, "Lot-sizing and throughput times in a job shop," *International Journal of Production Economics*, vol. 23, pp. 111–116, 1991.

[4] C. M. Barnhart, J. E. Wieselthier, and A. Ephremides, "Admission control policies for multihop wireless networks," *Wireless Networks*, vol. 1, no. 4, pp. 373–387, 1995.

[5] T. Ibaraki and N. Katoh, *Resource Allocation Problems: Algorithmic Approaches.* Cambridge, MA: MIT Press, 1988.

[6] R. Parker and R. Rardin, *Discrete Optimization.* Inc, Boston: Academic Press, 1988.

[7] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines.* New York, NY: Wiley, 1989.

[8] J. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: University of Michigan Press, 1975.

[9] D. Yan and H. Mukai, "Stochastic discrete optimization," *SIAM Journal on Control and Optimization*, vol. 30, pp. 549–612, 1992.

[10] W. B. Gong, Y. C. Ho, and W. Zhai, "Stochastic comparison algorithm for discrete optimization with estimation," *Proc. of 31st IEEE Conf. on Decision and Control*, pp. 795–800, 1992.

[11] L. Shi and S. Olafsson, "Nested partitions method for global optimization," *Operations Research*, vol. 48, pp. 390–407, 2000.

[12] Y. C. Ho, R. S. Sreenivas, and P. Vakili, "Ordinal optimization in DEDS," *J. of Discrete Event Dynamic Systems: Theory and Applications*, vol. 2, pp. 61–88, 1992.

[13] C. G. Cassandras, L. Dai, and C. G. Panayiotou, "Ordinal optimization for deterministic and stochastic resource allocation.," *IEEE Trans. Automatic Control*, vol. 43, no. 7, pp. 881–900, 1998.

[14] H. Kushner and D. Clark, *Stochastic Approximation for Constrained and Unconstrained Systems.* Berlin, Germany: Springer-Verlag, 1978.

[15] Y. C. Ho and X. Cao, *Perturbation Analysis of Discrete Event Dynamic Systems.* Dordrecht, Holland: Kluwer Academic Publishers, 1991.

[16] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems.* Kluwer Academic Publishers, 1999.

[17] C. G. Cassandras and V. Julka, "Scheduling policies using marked/phantom slot algorithms," *Queueing Systems: Theory and Applications*, vol. 20, pp. 207–254, 1995.

[18] J. Wieselthier, C. Barnhart, and A. Ephremides, "Standard clock simulation and ordinal optimization applied to admission control in integrated communication networks," *Journal of Discrete Event Dynamic Systems*, vol. 5, pp. 243–279, 1995.

[19] K. Gokbayrak and C. G. Cassandras, "Adaptive call admission control in circuit switched networks," *IEEE Transactions on Automatic Control*, 2000. Submitted.

[20] C. G. Cassandras and R. Yu, "A 'surrogate problem' approach for lot size optimization in manufacturing systems," *Proc. of 2000 American Control Conference*, pp. 3279–3283, 2000.

[21] C. G. Panayiotou and C. G. Cassandras, "Optimization of kanban-based manufacturing systems," *Automatica*, vol. 35, pp. 1521–1533, September 1999.