Università degli Studi di Roma Tre
Dipartimento di Informatica e Automazione
Via della Vasca Navale, 79 – 00146 Roma, Italy

A stochastic approach for the Lot Sizing problem

L. Adacher, , C. G. Cassandras

RT-DIA-81-2003 **Jun 2003**

adacher@dia.uniroma3.it,cgc@bu.edu

ABSTRACT

In this paper we extend the generalized surrogate optimization approaches and the stochastic comparison so as to tackle the lot-sizing problem in manufacturing system. In practice, with the surrogate methodology, the lot sizes are continuously adjusted on-line by the gradient-based approaches. The lot sizing determines the number of parts batched together for production. We utilize the queueing approach that evidences the existence of a convex relationship between batch size and waiting time (including processing). Large lot sizes will cause long lead times (the batching effect), as the lot size gets smaller the lead time will decrease but once a minimal lot size is reached a further reduction of the lot size will cause high traffic intensities resulting in longer lead times (the saturation effect). The congestion phenomenon is due to the increased number of setups (and thus total setup time). We comparison two different algorithms the "surrogate optimization" and the "stochastic comparison", and we show that the surrogate problem methodology recovers the optimal solution of the original discrete problem and exhibits very fast convergence compared to known discrete stochastic optimization methods. Some numerical results are reported.

1 Introduction

The design and control of *discrete event* and *hybrid systems* frequently involve discrete parameters. In designing manufacturing systems, for instance, the determination of buffer sizes and numbers of equipment of some type, fixture or pallets are crucial. Since such systems must usually be analyzed in a stochastic discrete optimization problems of considerable complexity due to the combinatorially explosive nature of the discrete search spaces involved. The problem of *stochastic optimization* for arbitrary objection functions presents a dual challenge. First, one needs to repeatedly estimate the objective function, when no closed form expression is available, this is only possible via simulation. Second, one has to face the possibility of determining local, rather the global optima. One common approach to solve the *continuous optimization problem* is based on estimated gradient information which drives the optimization process to a minimal point. However, this approach can easily lead to a local minimum, To overcome this problem, it is necessary to allow the optimization process to occasionally move to a bad neightboring point so as to provide the opportunity to jump out of a local minimum. While the area of stochastic optimization over continuous decision spaces is rich and usually involves gradient-based techniques as in several well-know stochastic approximation algorithms (e.g., [23],[19]), the literature in the area of *discrete stochastic optimization* is relatively limited. Most known approaches are based on some form of random search, with the added difficulty of having to estimate the cost function at every step. Such algorithms have been recently proposed by Yan and Mukai (e.g., [24]), Gong et al. (e.g., [13]). For this reason the *surrogate method* is proposed (e.g., [11],[10]), with this approach the original discrete set is trasformed into a continuous set over which a surrogate optimization problem is defined and subsequently solved. As in earlier work in (e.g., [2],[3]) an important feature of this approach is the every discrete state in the optimization process remains feasible, so that this scheme can be used on line to adjust the decision vector as operating conditions change over time. Thus, at every step of the continuous optimization process, the continuous state obtained is mapped back into a feasible discrete state; based on a realization under feasible state, new sensitivity estimates are obtained that drive the surrogate problem to yield the next continuous state. The proposed scheme, therefore, involves an interplay of sensitivity-driven iterations and continuous-to-discrete state trasformation. In practice, with the *surrogate methodology*, the discrete optimization is trasformed into a surrogate continuous problem where gradient based approach are used and the discrete problem solution is continuously adjust on line. Using this new methodology the construction of the neighbor is more simple and with a careful choose of the step size is possible to jump out of a local minimum.

In this paper, we extend a surrogate problem approach so as to tackle the *lot sizing problem* in manufacturing systems. The paper is organized as follows: In Section 2 we show the basic concepts of the stochastic optimization and the stochastic comparison algorithm is extend to our problem. Section 3 provides a framework for the surrogate approach. in Section 4 we introduce the lot sizing problem and we propose a formulation of the surrogate approach for this problem. In Section 5 we present some representative numerical results from application of those optimization approaches to the lot sizing problem.

2 The "Stochastic Comparison" optimization approach

For a stochastic system, let $L(\theta, \omega)$ denote a sample performance function, where θ is a parameter vector and ω is a sample path. For each given θ, let $E[L(\theta, \omega)]$ denote the mean of $L(\theta, \omega)$. For a complex stochastic systems, a closed form expression for $E[L(\theta, \omega)]$ in terms of the parameter θ is usually unavailable. Therefore, for each given θ, $E[L(\theta, \omega)]$ has to be estimated through simulation or direct observation of the system. For simplicity, set

$$J(\theta) = E[L(\theta, \omega)] ,$$

and let $\widehat{J}(\theta, \omega)$ represent an estimate of $J(\theta)$ based on a sample path ω. By repeated estimations $\widehat{J}(\theta, \omega)$ of $J(\theta)$ based on simulation or direct observation under direct values of θ, we are interested in finding an obtimal point θ^*, so that $J(\theta)$ is minimized. For this problem, there two major approaches. The first approach is based on using estimated derivative information of $J(\theta)$ to drive an iterative process to a minimal point. The simplest and the most common derivative estimation algorithm is called brute force (BF) method, that is the most general estimator, since it does not depend on any specific structure of the stochastic system under study. One the one hand, an appropriate choice for a first derivative estimator is very difficult to make. Obviously, for a general function $J(\theta)$ which has multiple local minima, the derivative optimization approach may ultimately settle down at such a local minimum. Therefore, even in the absence of estimation noise, several algorithms of the random search variety have been developed to overcome this problem. For example, the simulated annealing (SA) (e.g., [6], [1]) is very close to the derivative approach with one exception: it allows the optimization process to move to bad neighboring points (i.e., higher cost or uphill points) with a positive probability. In order to make sure that the optimization process will finally settle down at some close to the optimum, a control parameter called the "temperature" is used so that the probability of a bad move is gradually reduced to zero. Motivated by the stochastic rules (SR) algorithm (e.g., [24]), Gong et al. proposed a scheme refereed to as the stochastic comparison (SC) algorithm (e.g., [13]). This algorithm is aimed at discrete optimization problems with an objective function estimated through Monte Carlo simulation. It is shown to be more efficient in solving discrete optimization problems with estimation, since it performs a global search for the optimal point. A key feature of this algorithms is that it couples the estimation noise with a "cooling" strategy as following: instead of cooling down the randomization of the search sequence by the artificial function $\exp(-\Delta/T)$, the SC algorithm increases the number of estimated objective function comparisons gradually before each move has to be made. The following algorithm is the same that we have used to compare with the surrogate method:

For a finite set S, assume that the objective function $J(n)$, $n \in Z_+$ is a lot sizing vector, can be estimated by $\widehat{J}(\theta, \omega)$ via simulation. To obtain a good estimated we must simulate an elevate number of produced job by the system. Our objective is to determine lot sizes $n = [n_1, \ldots, n_N]^T$ so as to minimize the overall *mean system time* (or delay) of jobs, denoted by $J(n)$.

We define $\{M_k\}$, k=1,2,... to be non-decreasing integer sequence and assume that increasing logarithmically (a logarithmic increase is needed to guarantee convergence of the optimization process). M_k is the number of comparison made between estimates of the current state and a candidate state before the kth move is made

$R(i,j)$ is a preselected probability matrix, with $i, j \in \mathrm{Z}_+$. Here, $R(i,j)$ stands for the probability that the state j will be selected as the next state visited by the optimization process, given the current state is i.

1. Initialize: $k = 0$, select n^0

2. For given $n^k = i$, choose a next candidate state $\overline{n}^k = j$ from $\mathrm{Z}_+ - \{i\}$, with probability $R(i,j)$

3. $n^{k+1} = n^k$ with probability p_k and $n^{k+1} = \overline{n}^k$with probability $(1 - p_k)$

where $p_k = \{P[\widehat{J}(j,\omega) < \widehat{J}(i,\omega)]\}^{M_k}$

However, it is realizable in the following way: both $\widehat{J}(j,\omega)$, $\widehat{J}(i,\omega)$ are estimated M_k times. If $\widehat{J}(j,\omega) < \widehat{J}(i,\omega)$ every time, then we set $n^{k+1} = \overline{n}^k$, otherwise $n^{k+1} = n^k$. One the key features of the SC algorithm is the fact that the iteration steps are based on the estimated order of two objective function value, $J(n)$ and $J(\overline{n})$, not their cardinal values. Thus we exploit the fact that order statistics are generally very robust with respect to estimate noise In other word, estimating the order of two variables is much simpler problem than estimating their actual values (e.g., [16]). It follows that good moves can be made based on relatively poor estimates. This implies that short simulation runs are adequate, which means that one can aspect the SC algorithm converge quickly in general. In fact, in practice, one can see that the algorithm rapidly identifies points that are close to optimal, but getting to the actual optimum take very long time.

3 The 'Surrogate Problem' Optimization Approach

The surrogate problem methodology was initially developed in [11] for resource allocation problems of the form:

$$\min_{r \in A_d} J_d(r) \tag{1}$$

where r is an N-dimensional decision vector with r_i denoting the number of resources that user i is assigned subject to a capacity constraint

$$A_d = \{r : r = [r_1, \dots, r_N]', \sum_{i=1}^{N} r_i = K, r_i \in \mathrm{Z}_+\} \tag{2}$$

and $J_d(r) = E_\omega[L_d(r,\omega)]$, where $L_d(r,\omega)$ is the cost incurred when the state is r over a specific sample path denoted by ω. The integer capacity constraint is relaxed and a resulting surrogate problem is given by

$$\min_{\rho \in A_c} J_c(\rho)$$

$$A_c = \{\rho : \rho = [\rho_1, \dots, \rho_N] \tag{3}$$

The basic idea of this approach is to solve the continuous optimization problem above with standard stochastic approximation methods (e.g., [10]) and establish the fact that when (and if) a solution ρ^* is obtained it can be into some point $r = f(\rho^*) \in A_d$ which is in fact the solution of (1). Note, however, that the sequence $\{\rho_k\}$, $k = 1, 2, \ldots$ generated by an iterative scheme for solving (3) consists of real-valued allocations which are infeasible, since the actual system involves only discrete resources. Thus, a key feature of our approach is that at every step k of the iteration scheme involved in solving (3) the discrete state is updated through $r_k = f_k(\rho_k)$ as ρ_k is updated. 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 employed to obtain cost sensitivities from the *actual* operating system at every step of the process. 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. Related to the basic scheme initially , we set the "surrogate system" state to be that of the actual system state, i.e. $\rho_0 = r_0$. Subsequently, at the kth step of the process, let $H_k(\rho_k, r_k, \omega_k)$ denote an estimate of the sensibility of the cost $J_c(\rho_k)$ with respect to ρ_k obtained over a sample path ω_k of the actual system operating under allocation r_k. Two sequential operations are the performed at the kth step:

1. The continuous state ρ_k is updated through
$$\rho_{k+1} = \pi_{k+1}[\rho_k - \eta_k H_k(\rho_k, r_k, \omega_k)], \tag{4}$$
where $\pi_{k+1} : \mathrm{R}^N \to A_c$ is a *projection function* and η_k is a *step size* parameter.

2. The newly determined state of the surrogate system, ρ_{k+1}, is trasformed into an actual feasible discrete state of the original system through
$$r_{k+1} = f_{k+1}(\rho_{k+1}), \tag{5}$$
where $f_{k+1} : A_c \to A_d$ is a *projection mapping* of feasible continuous states to feasible discrete states.

One can recognize in the step 1 the form of stochastic approximation algorithm that generate a sequence $\{ \rho_k\}$ aimed to solve the surrogate problem. However, there is an additional operation, the step 2, for generating a sequence $\{ r_k\}$ which we would like to see converge to r^*. It is importance to note that $\{r_k\}$ corresponds to feasible realizable states based on which one can evaluate estimates $H_k(\rho_k, r_k, \omega_k)$ from observable data, i.e., a sample path of the actual system under r_k (not the surrogate state ρ_k). We can therefore see that this scheme is intended to combine the advantages of stochastic approximation type of algorithm with the ability to obtain sensitivity estimates with respect to discrete decision variables. We can summarize the results of the surrogate method as the following optimization algorithm for the solution of the discrete problem:

*Step*0. Initialize $\rho_0 = r_0$ and perturb ρ_0 to have all components non-integer. For any iteration $k = 0, 1, \ldots$

*Step*1. Determine $\mathcal{N}(\rho_k)$, the set of all *feasible neighboring discrete states of* ρ_k

$\mathcal{N}(\rho_k) = \{r | r = \inf \rho_k + \overline{\overline{r}}$ for all $\overline{\overline{r}} \in \{0, 1\}^N\} \cap A_d$.

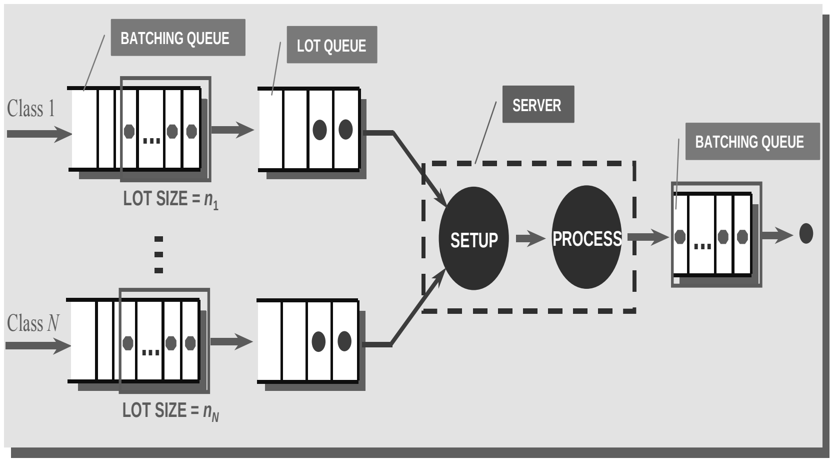

Figure 1: Modeling framework for the lot-sizing problem

.

*Step*2. Determine $\mathcal{S}(\rho_k)$, a selection set to define a set whose a convex hull includes ρ_k (using ([10])).

*Step*3. Select a *trasformation function* $f_k \in \mathcal{F}_{\rho_k}$ such that $r_k = f_k(\rho_k) = \arg\min_{r\in\mathcal{S}(\rho_k)} \|r - \rho_k\|$.

*Step*4. Evaluate the gradient estimation $\nabla L_c(\rho_k) = [\nabla_1 L_c(\rho_k), ..., \nabla_N L_c(\rho_k)]^T$, using the following relationship $\nabla_j L_c = L_d(r^j) - L_d(r^k)$, where k satisfies $r^j - r^k = e_j$ (see ([10])).

*Step*5. Update state: $\rho_{k+1} = \pi_{k+1}[\rho_k - \eta_k \nabla L_c(\rho_k)]$.

*Step*6. If some stopping condition is not satisfied, repeat steps for $k+1$. Else set ρ^*.

It is shown in [9], that the above algorithm converges to a global minimum under appropriate technical conditions (otherwise, it yields a local minimum).

4 The "Lot Sizing" Problem

In this paper, we consider the *lot sizing* problem in manufacturing systems.

A "lot" in manufacturing systems is a group of parts of similar type that are processed together at a workstation following a "setup" to accommodate this particular part type. When a lot has completed processing, the workstation switches over to a new part type through a setup associated with this new type. Clearly, during a setup the workstation is idle, so it is desirable to minimize the total setup time over a given production period. If lots are small, the workstation engages in frequent setups. If, on the other hand, lots are large, then part types must experience long queueing delays as they await their turn.

Speed has become a competitive requirement for almost any manufacturing organization. The focus is on timely delivery of quality products at the lowest possible cost.

Simultaneously the manufacturing concept has changed: make-to-stock policies are increasingly replaced by make-to-order (assembly-to-order) policies. Philosophies like Just-In-Time (JIT); time-based competition and concurrent engineering have amplified and supported this shift. Recently the idea of combining batching and queueing models has attracted great attention from many researchers, because it introduces the aspect of time from queueing into inventory theory. It is clear that queueing delays constitute a major part of the manufacturing lead-time, an aspect that is ignored in the classical inventory models. Models relating queueing delay to batching represent the production facility as a queueing system with one or more servers where orders (or batches) represent individual customers. We analyze a queue of orders in front of the production facility, but there is no finished goods inventory (once produced, products are immediately delivered to the customer). Karmarkar initiated pioneering work([17],[18]). The main result from this queueing approach is the existence of a convex relationship between batch size and waiting time (including processing). Large lot sizes will cause long lead times (the batching effect), as the lot size gets smaller the lead time will decrease but once a minimal lot size is reached a further reduction of the lot size will cause high traffic intensities resulting in longer lead times (the saturation effect). The congestion phenomenon is due to the increased number of setups (and thus total setup time). High traffic analysis is often used to describe the queueing behavior for high saturation levels. One of the major analytical problems with this queueing approach is the inclusion of the setup time. That is the reason why most models discussed in the literature assume that the individual customers are batches. The arrival of batches is modeled as Poisson process and exponentially distributed batch processing times (including a setup time) is assumed for the processing stage. Once the bulk process is assumed, standard results from queueing theory can be applied. The relaxation of the batch arrival and batch service assumption has been proven to be difficult. Lambrecht, Chen and Vandaele assume individual customer arrivals and focus on the sojourn time of individual customers. The model can be described as follows: assume a demand process, characterized by λ individual customer arrivals per time unit. Next we wait until N customers have arrived (batch collection; the batch size is the decision variable) after which the production facility incurs a setup using τ time units. Finally, the customers are served at a rate of μ customers per time unit on a first come, first served basis. Once the individual customer is served, the customers leave the system and have to wait until the complete batch is finished. We are interested in the time W that customers spend in the system (sojourn time). At first sight, this model looks straightforward, but unfortunately, it is not. None of the existing queueing results, neither batch arrivals nor batch service models can be used. The key problem is to find out how long customers stay in the batch queue phase. An individual arrival and departure process, with an intermediate bulk stage characterize technically the model, further complicated by a non-zero setup time.

The lot sizing problem is a complex optimization problem: lot sizes take integer values defining a large discrete state space, while the system itself operates in a stochastic environment where one can typically only estimate average part system times through direct observation or simulation (Karmakar [17]).

To place the problem in an appropriate modeling framework, Fig. 1 shows a workstation serving N different part types. We will refer to parts as *jobs* and to part types as *classes*. The interarrival and process time for each job of class i is a random variable with corresponding distributions G_a^i and G_s^i. The arrival and processing rates are denoted by

λ_i and μ_i respectively with $i = 1, \ldots, N$. Arriving jobs of class i join a queue where n_i of them are "batched" on a FCFS basis to define a lot. Once there is a lot waiting in the ith queue, the server can begin processing this class. We assume this is done according to a given schedule, so that the server visits classes $i = 1, \ldots, N$ in round-robin fashion as long as there is a lot available, otherwise service is provided to the next available class. When the server switches over to class i from any other class $j \neq i$, a setup time S_i is involved, which is generally also a random variable. For simplicity, we shall henceforth assume that S_i is a fixed setup time depending only on the class type. Thus, the mean service time of a class i job includes both setup and processing components and is given by

$$\frac{1}{\mu_{J_i}} = \frac{1}{\mu_i} + \frac{S_i}{n_i} \tag{6}$$

When a lot is served, jobs are processed individually and must wait until the entire lot is complete. Then, the lot is released to the next server (or exits the system). Thus, a large lot size causes delays due to waiting for batching at both the entrance and exit batching queues shown in Fig. 1.

Our objective is to determine lot sizes $n_1, \ldots, n_N$ so as to minimize the overall *mean system time* (or delay) of jobs, denoted by $J(n_1, \ldots, n_N)$. In what follows, we will assume that $n_1, \ldots, n_N$ are always selected so as to satisfy stability conditions (to be made explicit in the sequel) and that the systems considered satisfy standard ergodicity assumptions. A typical cost function with respect to a single lot size, say $J(n_i)$, is shown in Fig. 2. However, analytical expressions for such functional relationships, even under simple assumptions on the distributions of interarrival, service, and setup times, are unavailable. One typically resorts to methods for stochastic discrete optimization that involve some form of random search and estimation of the unknown cost function $J(n_1, \ldots, n_N)$ at different observed (or simulated) lot size settings.

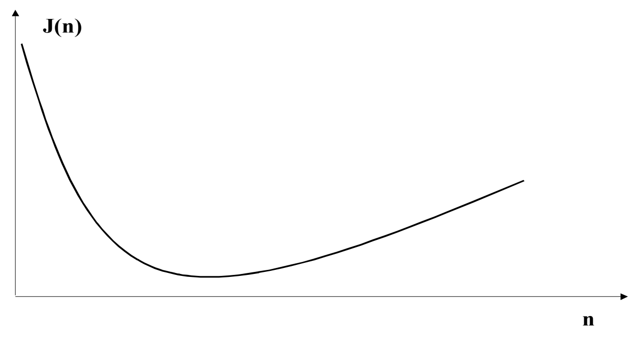

Figure 2: Typical cost function for the lot-sizing problem

In this paper, we tackle the lot sizing problem using an *on-line* optimization approach, i.e., we iteratively adjust the (integer) lot size parameters $n_1, \ldots, n_N$ for N part types based on data directly observed and aiming at minimizing the overall *mean system time* (or delay) of parts, denoted by $J(n_1, \ldots, n_N)$. Recently, we proposed an approach for solving such problems based on the idea of transforming a *discrete* optimization problem into a "surrogate" *continuous* optimization problem which is not only easier to solve, but also much faster using standard gradient-based approaches [10]. The two key issues related to this approach are (a) obtaining the actual solution of the original problem from the surrogate one, and (b) using this approach on-line, i.e., making sure that at every step of the iterative solution process a *feasible* discrete state is defined from an (infeasible) surrogate state.

The lot sizing problem can be formulated as follows:

$$\min_{n \in A_d} J_d(n) \tag{7}$$

where n is an N-dimensional decision vector with $n_i \in \mathrm{Z}_+$ denoting the class i lot size, and

$$A_d = \{n : n = [n_1, \dots, n_N]', \sum_{i=1}^{N} \lambda_i \left(\frac{1}{\mu_i} + \frac{S_i}{n_i} \right) < 1, 0 < n_i < K_i\} \tag{8}$$

the constraint set A_d contains a stability condition to ensure that the total number of jobs in the system remains finite w.p. 1. In addition, we assume that an upper bound K_i is imposed on each lot size parameter n_i. The corresponding "surrogate" problem in this case is obtained by relaxing the lot sizes to $\rho_i \in \mathrm{R}_+$ to obtain

$$\min_{\rho_i \in A_c} J_c(\rho)$$

$$A_c = \{\rho : \rho = [\rho_1, \dots, \rho_N]', \sum_{i=1}^{N} \lambda_i \left(\frac{1}{\mu_i} + \frac{S_i}{\rho_i} \right) < 1 \quad 0 < \rho_i < K_i\} \tag{9}$$

As explained in the previous section, the first step is to define the neighborhood set $\mathcal{N}(\rho)$ and then identify a selection set $\mathcal{S}(\rho)$ that contains $N+1$ linearly independent discrete neighboring states, denoted by n^j, $j = 1, \dots, N+1$, whose convex hull includes ρ. In this case, we need to extend the approach to include $N+1$ states, rather than N, because there is no capacity constraint $\sum_{i=1}^{N} \rho_i = K$ which allowed us to reduce the dimensionality of $\mathcal{S}(\rho)$ by 1. Thus, the constraints shown in (9) will be explicitly accounted for through an appropriate projection in the algorithm presented below. Because of this difference, the gradient is modified as follows:

$$\nabla L_c(\rho) = \begin{bmatrix} 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} e & \mathbf{R} \end{bmatrix}^{-1} L_d^{\mathcal{S}} \tag{10}$$

where $\mathbf{I}$ is the identity matrix of dimension N, e is an $N+1$ dimensional vector with all entries equal to 1, $\mathbf{R}$ is the matrix whose rows are $n^j \in \mathcal{S}(\rho)$, and $L_d^{\mathcal{S}} = [L_d(n^1), \dots, L_d(n^{N+1})]'$ is the vector of costs for these discrete states (which we may obtain using Perturbation Analysis (PA) and Concurrent Estimation (CE) ([15],[5]). The crucial aspect is the dermination of the set $\mathcal{S}(\rho_k)$; in fact the principal steps of the algorithm are based on this set (see [10]). Based on the above, the optimization scheme we use is the same proposed and the algorithm is the following:

First, initialize ρ_0. Then, for any iteration $k = 0, 1, \dots$:

1. Determine $\mathcal{N}(\rho_k)$ and $\mathcal{S}(\rho_k)$

2. Select $f_k \in \mathcal{F}_{\rho_k}$ such that $n_k = f_k(\rho_k) = \arg\min_{n \in \mathcal{S}(\rho_k)} \|n - \rho_k\|$

3. Evaluate $\nabla L_c(\rho_k)$ [using (10)]

4. Update state: $\rho_{k+1} = \pi_{k+1}[\rho_k - \eta_k \nabla L_c(\rho_k)]$.

5. If some stopping condition is not satisfied, repeat steps for $k+1$. Else, set ρ^*

Finally, we obtain n^* as one of the neighboring feasible states in the set $\mathcal{S}(\rho^*)$. In step 4, η_k is the step size and π_{k+1} is a projection such that if $\rho_{k+1} \in A_c$, then $\pi_{k+1}(\rho_k) = \rho_{k+1}$, otherwise, the projection maps ρ_{k+1} to a point in A_c which is closest to ρ_{k+1}. The projection mapping is a crucial element and may have a significant effect on convergence, for every current solution ρ_k the following steps are repeat:

There are two different types of constrains

a) Capacity constrains $\rho_i < K_i$
If there is a component i of the current vector solution ρ_k that is $\rho_i^k \geq K_i^k$ Then $\rho_i^{k+1} = K_i^k - eps$ (where K is the capacity and eps is a parameter of the algorithm)

b) Stability constrains $\sum_{i=1}^{N} \lambda_i \left(\frac{1}{\mu_i} + \frac{S_i}{\rho_i} \right) < 1$
To find the closet point in the feasible set A_c to the point ρ_k we considered only the ratio S_i/ρ_i^k(S_i is the set-up for the lot i and ρ_i^k is the component i of the current solution ρ_k) and we increment only the component that give the maximum value of this ratio

$i = argMax_j(S_j/\rho_j^k) \rightarrow \rho_i^{k+1} = \rho_i^k + eps$

5 Numerical results

Our objective is to determine lot sizes $n^* = [n_1^*, \ldots, n_N^*]$ so as to minimize the overall *mean system time* (or delay) of jobs, denoted by $J(n_1, \ldots, n_N)$. The arrival process is Poisson with rate λ_i with $i = 1, \ldots, N$, the service times of the server is exponentially distributed with rates μ_i with $i = 1, \ldots, N$. When the server switches over to class i from any other class $j \neq i$, a setup time S_i is involved We illustrate the surrogate method for 3-classes simulated model, in all cases we assume that lot size are constrained by $n_i < 100$ $i = 1...N$. Three different parameter settings were used, it is showed in Table 1

CASE	λ	μ	S
1	[4,4,4]	[20,40,60]	[1,0.5,0.7]
2	[4,4,8]	[20,40,80]	[1,0.5,0.7]
3	[4,6,8]	[20,40,80]	[1.0,1.5,1.7]

Table 1: Parameter values

We use the constant step size sequence $\eta_k = 10.5$ and constant observation intervals ($number\ of\ job\ completed = 900.000$). The corresponding results for the surrogate method were $n^* = [19, 15, 16]$, $n^* = [20, 17, 29]$ and $n^* = [34, 50, 68]$ for the three cases respectively.

The following table shows that different start points converge at the same optimal solution for the *case2*

START POINT	STEP	SOLUTION n	J$_d(n)$
$[50.95, 74.28, 34.65]$	281	$[20, 17, 29]$	4.11
$[59.54, 50.39, 35.48]$	267	$[20, 17, 29]$	4.11
$[22.29, 48, 28, 20.35]$	138	$[20, 17, 29]$	4.11
$[35.12, 78.70, 20.14]$	282	$[20, 17, 29]$	4.11
$[52.59, 57.80, 16.99]$	232	$[20, 17, 29]$	4.11
$[46.68, 5.72, 7, 63]$	249	$[20, 17, 29]$	4.11
$[34.7, 17.42, 27.42]$	259	$[20, 17, 29]$	4.11

Table 2: Example: different start points that converge at the optimal solution

Simulating many different start points we have seen that the algorithm converges in different point really close, but it doesn't converge only in one. The following table shows the range of convergence considering 50 different start points, that are generated in random way for the *case 2*

SOLUTION n	J$_d(n)$
[20,17,29]	4.117
[19,16,32]	4.124
[19,18,30]	4.121
[22,17,28]	4.129
[22,16,29]	4.127

Table 3: Range of convergence

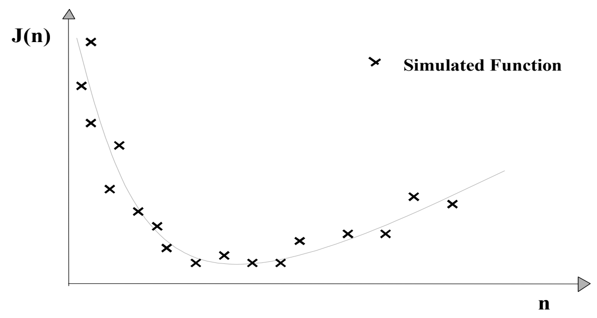

Figure 3: Simulated cost function for a lot-sizing problem

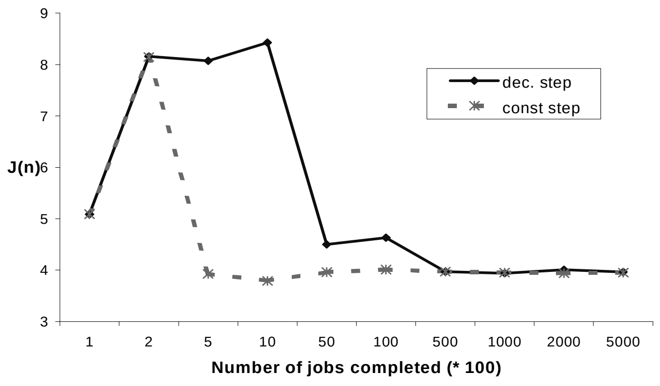

Figure 4: Convergence of the surrogate method

The reason for this behavior is related with the structure of the system, in fact the closed form expression for $J(n_1, \ldots, n_N)$ is unavailable, and it is estimated through simulation of the stochastic system. In Fig. 3 is showed the form of cost function for a lot-sizing problem obtained via simulation.

We have simulated the same cases with the stochastic comparison method, and the SC method give always a solution in the "range of convergence" of the surrogate method, it is show in Table 4 for the *case1.* These solutions are obtained comparison $\sum_i^k 2^i$different points, with $k = 20$, and the *mean system time* of jobs is simulated with a constant observation intervals (*number of job completed* $= 200.000$).

RANGE OF CONV.	STOCHASTIC C.
[17,16,16] J(n)=3.943	[17,16,16] J(n)=3.943
[18,13,18] J(n)=3.938	
[16,14,19] J(n)=3.939	
[19,15,16] J(n)=3.935	
[16,16,18] J(n)=3.947	

Table 4: Comparison between SM and SC

How we showed in the table 4 the surrogate method and the stochastic comparison practically find the same solution ([17,16,16], J(n)=3.943). To show how the surrogate method converge, we consider a start point far from the optimal solution. In Fig.4 we show the convergence of the mean time system when the number of jobs produced increase. We considered two different step size to show that for the lot sizing problem the constant step size is better. The system is started with a initial allocation [74,6,70], and the step size sequences are $\eta_k = 10.5$ (constant step) $\eta_k = 20/(k+1)$ (decreasing step), the algorithm performs as follows:

6 Generalization

In this section, we analize a generalization of the above system , now we consider a system with two different one server machines in line.To place the problem in an appropriate modeling framework, Fig. 5 shows a two workstations serving N different part types. The arrival at the first machine rate is denoted by $\lambda_i$and processing rates are denoted by μ_i with $i = 1, \ldots, N$ and $i = N+1, \ldots, 2N$ for the first and the second machine respectively. Arriving jobs of class i join a queue where n_i of them are "batched" on a FCFS basis to define a lot. Once there is a lot waiting in the ith queue, the server 1 can begin processing this class. When a lot is served, jobs are processed individually and each single job wait in the next queue until the entire new lot $\underline{n_i}$ is complete and it will be processed by the server 2. Then, only the whole lot is released to exits the system. We assume that the servers proces the lots according to a given schedule, so the server visits classes $i = 1, \ldots, N$ in round-robin fashion as long as there is a lot available, otherwise service is provided to the next available class. When a server switches over to class i from any other class $j \neq i$, a setup time S_i is involved, which is generally also a random variable. For simplicity, we shall henceforth assume that S_i is a fixed setup time depending only on the class type.

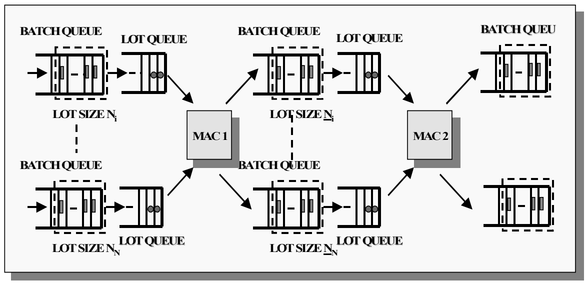

Figure 5: Modeling of system with two machines in line

Our objective is always to determine lot sizes $n_1, \ldots, n_N, \underline{n_1}, \ldots, \underline{n_N}$ so as to minimize the overall *mean system time* (or delay) of jobs, denoted by $J(n_1, \ldots, n_N, \underline{n_1}, \ldots, \underline{n_N})$. In what follows, we will assume that $n_1, \ldots, n_N, \underline{n_1}, \ldots, \underline{n_N}$ are always selected so as to satisfy stability conditions and that the systems considered satisfy standard ergodicity assumptions.

6.1 Numerical results

The arrival process is Poisson with rate λ_i with $i = 1, \ldots, N$, the service times of the server is exponentially distributed with rates μ_i with $i = 1, \ldots, N$. When the server switches over to class i from any other class $j \neq i$, a setup time S_i is involved We illustrate the surrogate method for 2-classes simulated model, in all cases we assume that lot size are constrained by $n_i < 100$ $i = 1 \dots N$. Two different parameter settings were used, it is showed in Table 5; in case1 we present small set-up time and in case2 big set-up times.

CASE	λ	μ	S
1	[3,4,6,15,11,19]	[22,19,25,69,47,80]	[0.09,0.1,0.2,0.1,0.07,0.09]
2	[3,4,6,15,11,19]	[22,19,25,69,47,80]	[1.09,2.1,1.2,2.1,2.07,1.9]

Table 5: Parameter values

We want to put in evidence the rule of the set-up times in the more complex systems. Only with two machines in line the objective function becomes more complicated, in fact there are more local minimum. For these two different setting, we find the better solution when we force the solution to have the same structure for the two different machines (i.e. we introduce the condition that the lots size of the second server is the same of the first server). The results are reported in Table 6, also the solutions found with the stochastic comparison are reported.

CASE	Stochastic C.	$J_d(n)$	Surrogate M.	$J_d(n)$	Same lot-size	$J_d(n)$
1	[3,8,8,20,8,6]	2.58	[5,5,8,15,6,12]	2.43	[4,5,11,4,5,11]	1.63
2	[30,87,47,30,87,55]	18.2	[59,69,36,43,69,42]	20.1	[42,54,55,42,54,55]	14.8

Table 6: Algorithms comparison

In this generalization, simulating many different start points we have seen that the algorithm converges in different point not really close, this behavior puts in evidence the existence of many different local minimum. To exceed this problem we have analyzed many different start points and than we check the best solution.

7 Conclusion

In this paper we applied the generalized "Surrogate Methodology" to the lot-sizing problem. Our results shown the good construction of the selection set $\mathcal{S}(\rho)$, that is the most crucial aspect of this approach. For the sample application, only one server in the system, the results are really satisfactory, but for the system with two different one server machines in line,to find a satisfactory solution we must considered many different start points. Naturally, with more complicated systems, there are the possibility to stop in a local mimima. It is interesting to understand the structure the solution considering the set-up values (i.e. when is better to consider different lot-size for the different machines), this is unedr study.

References

[1] E. Aarts and J. Korst, (1989). Simulated Annealing and Boltzmann Machines, *Wiley*, New York, NY.

[2] Cassandras C. G., Dai L., Panayiotou C. G, (1998). Ordinal Optimization for Deterministic and Stochastic Resource Allocation, *IEEE Trans. Automatic Control*, vol. 43, n. 7, (881-900).

[3] C. G. Cassandras and V. Julka, (1993). A New Approach For Some Combinatorially Hard Stochastic Optimization Problems, *Proc. of 31st Annual Allerton Conference on Communication, Control, and Computing*, (667-676).

[4] C. G. Cassandras and C. G. Panayiotou, (1999). Concurrent Sample Path Analysis of Discrete Event Systems, *Journal of Discrete Event Dynamic Systems: Theory and Applications"*,vol.9, 171-195

[5] C. G. Cassandras, (1993). em Discrete Event Systems: Modeling and Performance Analysis,Irwin Publ., Homewood, IL.

[6] S. B. Gelfand, and S. K. Mitter, (1989). Simulated Anneling with Noisy or Imprecise Energy Measurements, Journal of Optimization Theory and Application, vol 62, (49-62).

[7] E. G. Gladyshev, (1965). On Stochastic Approximation, em Theory of Probability and its Applications, vol. 10(2), (275-278).

[8] P. Glasserman, (1991).*Gradient Estimation via Perturbation Analysis*, Kluwer Academic Pub.

[9] K. Gokbayrak and C. G. Cassandras, (2001). An On-Line 'Surrogate Problem' Methodology For Stochastic Discrete Resource Allocation Problems, *Journal of Optimization Theory and Applications*, vol. 108, n. 2, (349-376).

[10] K. Gokbayrak and C. G. Cassandras. A Generalized 'Surrogate Problem' Methodology For On-Line Stochastic Discrete Optimization, *Journal of Optimization Theory and Applications*, Submitted 2001.

[11] K. Gokbayrak and C. G. Cassandras,(1999). Stochastic Discrete Optimization Using a Surrogate Problem Methodology, *Proceedings of 38th IEEE Conf. On Decision and Control*,(1779-1784).

[12] W. B. Gong and Y. C. Ho, (1987). Smoothed Perturbation Analysis of Discrete-Event Dynamic Systems, *IEEE Transactions on Automatic Control*, vol. (AC-32, 10),(858-866).

[13] W. B. Gong and Y. C. Ho and W. Zhai, (1992). Stochastic Comparison Algorithm for Discrete Optimization with Estimation, *Proc. of 31st IEEE Conf. on Decision and Control*, (795-800).

[14] Y. C. Ho, (1997). On the Numerical Solutions of Stochastic Optimization Problems,*IEEE Trans. Automatic Control*, vol. 42,(727-729).

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

[16] Ho Y. C., Sreenivas R., Vakili P., (1995). Ordinal Optimization of Discrete Event Dynamic Systems, *J. of Discrete Event Dynamic Systems: Theory and Applications*, vol. 2, (61-88).

[17] U.S. Karmakar, S Kekre and S. Freeman, (1985). Lot-sizing and lead-time performance in manufacturing cell, Interface, vol. 15, (1-9).

[18] U.S. Karmakar, (1987). Lot-sizing, lead-time performance in process inventorories, *Management Science*, vol. 33, (409-418).

[19] Kiefer J., Wolfowitz J., (1952). Stochastic Estimation of the Maximum of a Regression Function, *Annals of Mathematical Statistics*, vol. 23, (462-466).

[20] Kimms A., (1997). *Multy- Level Lot Sizing and Scheduling*, Phisica-Verlag.

[21] D. E. Kirk, (1970). *Optimal Control Theory*, Prentice-Hall

[22] H.J. Kushner and D.S. Clark,(1978). *Stochastic Approximation for Constrained and Unconstrained Systems*, Springer-Verlag.

[23] Robbins H., Monro R.,(1951). A Stochastic Approximation Method,*Annals of Mathematical Statistics*, vol. 22, (400-407).

[24] D. Yan and H. Mukai, (1992). Stochastic Discrete Optimization, *SIAM Journal on Control and Optimization*, vol. 30, (549-612).