Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers and Chemical Engineering 29 (2005) 1317–1328
Computers & Chemical Engineering
www.elsevier.com/locate/compchemeng

Simulation-based optimization with surrogate models—Application to supply chain management

Xiaotao Wan, Joseph F. Pekny*, Gintaras V. Reklaitis

School of Chemical Engineering, Purdue University, West Lafayette, IN 47906, USA

Available online 25 April 2005

Abstract

Simulation is widely used in the decision-making processes associated with supply chain management. In this paper, we present an extension of the simulation-based optimization framework which has been previously proposed for analyzing supply chains. The extension consists of the iterative construction of a surrogate model based on systematically accumulated simulation results to capture the causal relation between the key decision variables and supply chain performance. The decision variables can then be optimized using the surrogate model in place of individual simulation runs to economize on the overall computational effort. Several techniques are embedded in the framework to achieve the targeted objective: least square support vector machine (LSSVM), Bayesian evidence framework, and design and analysis of computer experiment (DACE). The extended framework is illustrated using two small examples and then applied to optimize the inventory levels in a three-stage supply chain. The results show that the framework identifies good solutions efficiently, can accommodate chance constraints and scales up well.

Keywords: Simulation-based optimization; Supply chains; Surrogate model; Support vector machine; Design and analysis of computer experiment; Safety stocks

1. Introduction

Monte Carlo simulation is one of the most important tools for analyzing supply chains in the presence of uncertainties. Compared to analytical techniques, simulation provides the flexibility to accommodate arbitrary stochastic elements, and generally allows modelling of all of the complexities and dynamics of real world supply chains without undue simplifying assumptions. However, as a descriptive method, simulation can only be used to perform optimization through "what if" case studies involving the comparison of several cases or scenarios.

The "what if" case approach is particularly ineffective in optimizing continuous decision variables (for example, inventory levels of each entity in a chain). Thus, having a method to perform optimization efficiently in continuous decision spaces is of importance. Even for scenario analysis, such a method is a necessary because inventory optimization generally appears as a sub-problem within each scenario. For instance, consider the problem of determining the best network design from several network candidates. One performance criterion for assessing the quality of a design is to evaluate the material hold-up under this particular design. To accomplish this, an inventory level optimization problem must be solved.

Optimization in continuous decision space can be carried out through simulation-based optimization (Fu, 2002). Under this framework, an optimization module is coupled with a simulation model. The optimizer searches the decision space systematically using the simulation model as function evaluation. While this basic framework is well understood, simulation-based optimization is not yet widely used in supply chain analysis practice mainly because of its extremely high computational requirements, though there are some recent attempts (Jung, Blau, Pekny, Reklaitis, & Eversdyk, in press; Subramanian, Pekny, & Reklaitis, 2000; Subramanian, Wan, Pekny, & Reklaitis, 2003). Typically, the optimization search process is slow and inefficient due to the presence of

* Corresponding author.
E-mail addresses: wanx@ecn.purdue.edu (X. Wan), pekny@purdue.edu (J.F. Pekny).

0098-1354/$ – see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2005.02.018

statistically and numerically caused noise in the simulation output as well as the fact that each simulation run itself takes significant execution time. However, simulation-based optimization provides an alternative, sometime even a better option for supply chain optimization problems where analytical methods dominate.

Among those widely studied simulation-based optimization algorithms (Fu, 2002), response surface methodologies (RSM) (Nedderneijer, van Oortmarssen, Piersma, & Dekker, 2000) and stochastic approximation (SA) (Kleinman, Spall, & Naiman, 1999) play important roles. RSM fit a local first- or second-order regression surface to guide search by removing the noise contained in the simulation results and providing a decent direction (assume minimization from now on) along which the objective function decreases. However, RSM are limited by their lack of flexibility to describe local surfaces that cannot be sufficiently approximated with a first- or second-order function (Greenwood, Paul Rees, & Siochi, 1998). Consequently, no successful application of RSM to relatively high-dimensional problems is reported. SA bears great similarity with nonlinear optimization, as it computes local gradients by perturbing the current decision points and uses the gradients to guide search. The convergence of SA is theoretically guaranteed, which is one of the primary reasons that make SA attractive. In this family, simultaneous perturbation stochastic approximation (SPSA) (Spall, 1992, 1999) is especially noteworthy; it computes the gradients with only two simulation runs independent of the dimensionality of the underlying problem while ordinary SA uses finite difference to compute gradients which incurs much more simulation runs as the function evaluation (i.e. simulation) increases linearly with the dimensionality. Amazingly, SPSA not only does not deteriorate the solution quality, it even leads to better solutions than SA, given that the total simulation effort is equal.

Another popular simulation-based optimization is meta-modelling or surrogate model building (Barton, 1994) where our proposed method belongs to. The essential idea is to fit a single surface for the whole decision space, and use this surface to perform optimization instead of the simulation model. The common practice simply uses a neural network to fit the desired surface (Chambers & Mount-Campbell, 2002; Sabuncuoglu & Touhami, 2002); much research yet to be done to address several apparent issues impacting the solution quality as well as the computational effort: how to avoid over- and under-fitting, how to capture complex casual relations, how to utilize available information to reduce simulation effort, how to handle high-dimensional problems, etc. Some of those questions are dealt in our research.

This paper presents an extension to the concept of simulation-based optimization by introducing a surrogate model together with experimental design and domain reduction for use in optimizing continuous supply chain decision variables, with the aim of mitigating the computational burden of existing methods. Comparing with the related previous work where special attention is paid on the inner loop optimization and the outer loop optimization is either simplified or problem specific (Jung et al., in press; Subramanian et al., 2000), this research focuses on developing a general and efficient algorithm for outer loop optimization based on the surrogate model principle, in which least square support vector machine (LSSVM) captures the casual relations embedded in simulation results, Bayesian evidence framework overcomes over- and under-fitting, while adaptive experimental design dynamically utilizes the accumulated information. In the following, Section 2 describes the proposed framework; Section 3 validates the framework; Section 4 applies the framework to case studies and conclusions are drawn in Section 5.

2. Simulation-based optimization framework

An overview of the proposed extended simulation-based optimization framework is shown in Fig. 1. As pointed above, unlike classical response surface methodologies which build local surfaces, the surrogate model fits a single surface for the whole domain. Essentially, this framework iteratively builds a series of temporary surrogate models (or meta-models) by gradually increasing the number of sampling points in the decision space, and uses these temporary models to guide the incremental sampling. The iterative refinement process is terminated when a suitable stopping criterion is met. The surrogate models are constructed with LSSVM, which has remarkable properties to capture even very complex input and output relations by discriminating the true signal from the noise without over-fitting the noisy simulation results. The incremental sampling is carried out using design and analysis of computer experiment (DACE) so as to balance exploration of the decision space against the exploitation of the already extracted model information. A domain reduction technique

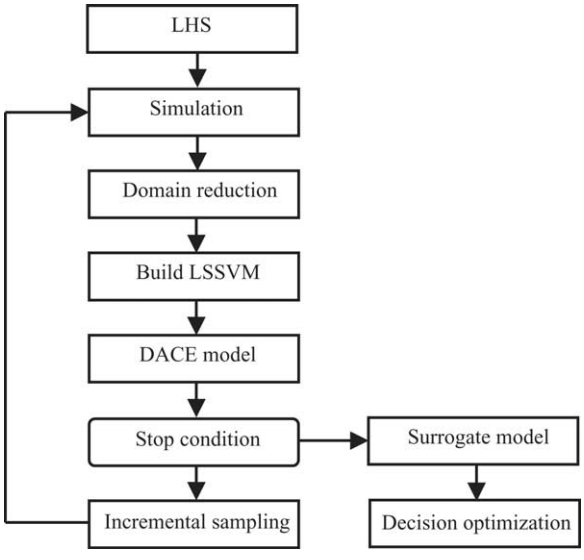

Fig. 1. Simulation-based optimization framework.

is also incorporated in each step to reduce the complexity of the surrogate models.

2.1. Sampling

The initial sampling is performed with the Latin hypercube sampling (LHS) technique. LHS disperses the sampling points as evenly as possible in the whole domain, thus providing a good basis for building the first temporary surrogate model. However, the property of the underlying response surface, which the surrogate model tries approximate, may varies widely across different regions of the domain. Some regions (for example, regions whose performance values have abrupt change) may need more sample points to reflect their structures, and others regions (for example, regions with relative flat response surfaces) only need a few samples. LHS is unable to differentiate these situations; it treats the whole domain homogenously. Other methods like Latin supercube sampling (LSS) (Owen, 1998) essentially follow the same line of logic as LHS, they try to evenly distribute the sampling without incorporating any information of the target response surface.

2.2. Domain reduction

The number of points that must be sampled and the complexity of the surrogate model are roughly proportional to the size of the domain. Domain reduction can significantly decrease the overall computational burden by cutting off the regions where good solutions are unlikely to exist and concentrating on exploring the promising regions. The solution quality can also be improved as a smaller domain generally implies a simpler input and output relation, and the simpler relation is more likely to be captured by a surrogate model with fidelity than a more complex relation. Domain reduction is achieved by comparing the average performance of different regions through fitting a regression tree to the simulation results which decomposes the domain into rectangular regions and computes an average performance value for each region. The actually algorithm is as follows:

Step 1: Build a regression tree for the noisy data, find the best leaf node which has the smallest average performance value.
Step 2: Starting from the best leaf node, trace up the tree to locate a promising region; denoting μ_c and σ_c as the average performance value and standard deviation of the current node, μ_s and σ_s as the corresponding values of the sibling node, move up the tree if $|\mu_s - \mu_c| < 3(\sigma_s + \sigma_c)$, otherwise return the current node as the promising region.

By assuming that regions with small average values have higher probability to contain good solutions than regions with large average values, further sampling is restricted to the promising region where good solutions are more likely to exist and consequently the surrogate model is only built for this region. Although there is no mechanism to prevent domain reduction from cutting off some good solutions even the global optimum, it does dramatically reduce computational burden and enhance the performance as long as the objective of the optimization is to find one good solution instead of unrealistically insisting to locate the optimum which is almost impossible for complex systems with various uncertainties.

2.3. Model construction

One of the crucial decisions in the proposed framework is to use LSSVM (Van Gestel et al., 2002) to construct the surrogate models. LSSVM has two important features: it is capable of capturing complex input and output relations as well as efficiently extracting structure information from noisy data.

LSSVM fits a linear model of the following form:

$$y = w'\varphi(x) + b + e \tag{1}$$

where $\varphi(x)$ maps a finite-dimensional vector x to an infinite-dimensional vector $\varphi(x)$, b is a constant and e is a normal distributed random variable. Essentially, LSSVM is a linear regression in an infinite-dimensional space with $w'\varphi(x)$ as the dot product of two infinite-dimensional vectors and b as the intercept. The map $\varphi(x)$ can be regarded as an infinite number of basis functions $[\varphi_1(x), \varphi_2(x), \ldots, \varphi_p(x)]'$, where $\varphi_i(x)$ are single-valued functions and $p \to \infty$. Thus, $w'\varphi(x)$ is equivalent to linear combination of an infinite number of basis functions, and proper selection of $\varphi(x)$ ensures that almost arbitrary function relations can be captured by LSSVM.

The ability of LSSVM to extract structure information from noisy data relies on the structure risk minimization of statistical learning theory (Vapnik, 1998, 1999). According to the theory, to identify the intrinsic structure of noisy data, it is insufficient to minimize the empirical risk, defined as the difference between model prediction and observed results, since the empirical risk can always be reduced to 0 by fitting a very complex model like a neural network, and to ensure that noise is not fitted as structures, the structure risk (or the complexity of the model) must be controlled. Consequently, the parameters w and b in (1) are obtained through solving the optimization problem:

$$\min \sum_{i=1}^{n} (y_i - w'\varphi(x_i) - b)^2 + \gamma w'w \tag{2}$$

The two terms in (2) represent the empirical risk and the structure risk, respectively, and the penalty factor γ controls the trade-off between the two risks. The first term, or the empirical risk, is simply the residual error; the second term, or the structure risk, is proportional to the norm of w: based on statistical learning theory, w with a larger norm corresponds to a more complex function than that with a smaller norm, and a complex function is more likely to over-fit noise. Since minimizing the empirical risk aims to avoid under-fitting (large residual error) and minimizing the structure risk aims to avoid over-fitting, the two aspects of LSSVM ensure that structure

information embedded in noisy data can be successfully extracted.

Solving (2) after observing n pairs of noisy data (x_i, y_i) leads to:

$$y(x) = \sum_{i=1}^{n} \alpha_i \varphi'(x)\varphi(x_i) + b = \sum_{i=1}^{n} \alpha_i K(x, x_i) + b \tag{3}$$

The replacement of the dot product $\varphi'(x_i)\varphi(x_j)$ with a kernel function evaluation $K(x_i, x_j)$ is based on Mercer's theorem (Schölkof & Smola, 2002). The introduction of a kernel function eliminates the need to specify the map $\varphi(x)$ because only the dot product $\varphi'(x)\varphi(x_i)$ is relevant in the LSSVM solution instead of $\varphi(x)$ itself.

In the proposed framework, radial basis function (RBF) is selected to be the kernel function. If the distance between two points x_i and x_j in a d-dimensional space is defined as:

$$d(x_i, x_j) = \sum_{h=1}^{d} \theta_h (x_i^h - x_j^h)^2 \tag{4}$$

then the RBF kernel is:

$$K(x_i, x_j) = \sigma^2 \exp(-d(x_i, x_j)) \tag{5}$$

The distance defined by (4) is not a Euclidean distance; it is a weighted sum of squares for every direction h. The kernel function defined by (5) essentially prescribes that the function value of two arbitrary points decreases exponentially according to their distance. Thus, based on (3), the performance value $y(x)$ of a decision point x is determined by its distances to the n observed points x_i; or equally speaking, the observed performance values y_i are generalized to unobserved point x based on the distances since y_i implicitly determine both θ_h and α_i as well as all other relevant parameters. The weights θ_h effectively controls how the generalization is performed along each direction: if θ_h is large, y_i are only generalized to a small neighbourhood of x_i along direction h because $K(x, x_i)$ approaches 0 rapidly as the distances between x and x_i increase along direction h; similarly, if θ_h is small, y_i are generalized to a large neighbourhood along direction h because $K(x, x_i)$ decrease slowly as the distance increase along direction h.

Before solving (2) to obtain α_i and b in (3), the parameters θ_h of RBF kernel and the penalty factor γ in (2) must be determined. Those parameters, generally called hyper-parameters in contrast to α_i and b, are of paramount importance for LSSVM to successfully extract the structure information from noisy data. The significance of θ_h has already been elaborated above. As for γ, if γ is too large, the structure risk in (2) is over-penalized and simple functions are preferred which leads to under-fitting and important structures may fail to be captured; if γ is two small, the structure risk is under-penalized and complex functions are preferred which leads to over-fitting and noise is treated as structure. Either case will inevitably cause bad performance of LSSVM, thus γ must be set properly. A Bayesian evidence framework (Mackay, 1992, 1995) has been adapted to infer the hyper-parameters of LSSVM from observed noisy data (Van Gestel et al., 2002). Under this framework, three levels of optimization are performed: the highest level optimizes parameters of the RBF kernel, the second level locates proper γ, and the first level obtains α_i and b; full details can be found in the cited references.

2.4. DACE model and incremental sampling

With a temporary surrogate model, the noise in simulation data is essentially removed because the LSSVM model gives a deterministic performance prediction at each point in the decision space, where the prediction approximates the expected performance value of the corresponding decision point. With this observation, the DACE model (Sacks, Welch, Michell, & Wynn, 1989), originally developed for deterministic computer experiments, is introduced to perform experimental designs. Basically, DACE model is fitted for the prediction of LSSVM and captures all the information LSSVM extracted from the noisy data, and the information is then utilized to guide incremental sampling. Thus, in this section, when y_i is referred, it implies the performance of x_i predicted by LSSVM instead of the noisy simulation result.

2.4.1. DACE model

In the proposed framework, DACE fits the following model for deterministic dataset (x_i, y_i):

$$y(x) = \mu + Z(x) \tag{6}$$

where μ is a constant and $Z(x)$ is a spatial (or equivalently multi-dimensional) Gaussian process with variance σ^2. According to (6), the DACE model $y(x)$ is actually a Gaussian process with mean equal to μ and covariance determined by $Z(x)$. One puzzle needs to be clarified before proceeding further: how can one expect a Gaussian process, which is random at every point, to correctly fit absolutely deterministic data? Perhaps the best way to explain this paradox is to take a Bayesian point of view: the uncertainty and randomness in (6) reflect our belief about the data, not the intrinsic property associated with the data. Before observing any y_i, our knowledge about the whole domain is uncertain; however, we may believe each point is more likely to take values in some ranges based on our prior experience or purely subjective guess. This belief is described as a Gaussian process under DACE: every performance value y_i follows a normal distribution $N(\mu, \sigma^2)$ and is correlated with other performance values, where the magnitude of σ^2 reflects the level of uncertainty in our belief. Since the data are deterministic, if the performance value y_i is observed, the knowledge about x_i is complete and we must have $\sigma(x_i) = 0$, i.e. no uncertainty in our belief for this observed point. Moreover, due to the correlation among each point in the domain, our belief about other unobserved points is automatically updated; unobserved points still follow normal distributions but will have different means and variances

from our initial specification or guess. What happens here is that, observing y_i not only provides complete information for x_i, it also provides partial information for other points; the partial information permits generalization of observed results to unexplored points, and the level of uncertainty for every point is reduced: the closer to observed points, the more certain they will be. For example, if we believe all the points are positively correlated with x_i (let us ignore other observed data temporarily), observing a positive y_i basically indicates that other points should also take positive values.

Let $\boldsymbol{y}$ denote the vector composed by y_i, $\boldsymbol{x}$ denote the vector composed by x_i, R denote the covariance matrix of $\boldsymbol{x}$, r denote the covariance vector between $\boldsymbol{x}$ and an arbitrary unobserved point x, and $\mathbf{1}$ denote the vector of all 1 with comfortable dimension. Solving (6) leads to (Jones, Schonlau, & Welch, 1998):

$$E(y|\boldsymbol{y}) = \tilde{\mu} + r' R^{-1}(\boldsymbol{y} - \mathbf{1}\tilde{\mu}) \tag{7}$$

$$\mathrm{var}(y|\boldsymbol{y}) = \sigma^2 \left[1 - r' R^{-1} r + \frac{(\mathbf{1}R^{-1}r)^2}{\mathbf{1}' R^{-1}\mathbf{1}}\right] \tag{8}$$

To simplify the notation, let $\tilde{y}(x)$ denote $E(y|\boldsymbol{y})$ and $s^2(x)$ denote $\mathrm{var}(y|\boldsymbol{y})$. Eq. (7) shows that with the information provided by observed data $\boldsymbol{y}$, the prediction for an arbitrary point is updated from the initial guess $\tilde{\mu}$; the extra second term in the prediction represents the information contributed by observed data which basically is the linear combination of the error terms $\boldsymbol{y} - \mathbf{1}\tilde{\mu}$ at sampled (or observed) points and the combination coefficients are decided by the covariance relations. Eq. (8) shows that three terms contribute to the prediction variance: the first term is just the original prior variance before any sampling; the second term represents the reduction of variance due to information provided by observed data; the third term adds extra uncertainty because the parameters $\tilde{\mu}$ is estimated instead of knowing its true value. As the third term is always smaller than the second term, variance reduction is ensured after observing some data, which obviously make sense. It is easy to verify with (7) and (8) that $\tilde{y}(x_i) = y_i$ and $s^2(x_i) = 0$. That is, at sampled points x_i, the prediction is equal to the observed value y_i and there is no uncertainty at all, which again is correct because there is no noise in the observed data and the fitted surface must pass all of them. Eq. (8) also indicates that uncertainties always exist for unobserved decision points in the form of variances, and the uncertainties increase as the distance to observed points increase since both covariance r and R decrease with increasing distance. This piece of information, i.e. the variances of unobserved points, provides the base to perform experimental design.

2.4.2. Incremental sampling via experimental design

Two design criteria are considered: maximizing Bayesian information and maximizing expected improvement (Jones et al., 1998; Sasena, Papalambros, & Goovaerts, 2002). Sampling based on maximizing Bayesian information basically minimizes the overall uncertainty about the decision space, and the decision space is explored globally and evenly, similar to LHS but with the information provided by the LSSVM model (or previous sampling). Sampling based on maximizing expected improvement focuses on the promising regions that may improve the current best solution identified by the temporary LSSVM model, thus exploits the existing information aggressively and search locally (its global search capability is weak though exists). By combining the two criteria together, the incremental sampling is controlled such that both global and local samplings are considered with emphasis being gradually shifted form global exploration to local exploitation.

2.4.2.1. Maximizing Bayesian information. Bayesian information (BI) is a measure quantifying the overall uncertainty of our belief about the decision space. For a continuous space, BI is defined as the expected entropy $E[-\log p(y)]$, where $-\log p(y)$ is the entropy for each point and the dependence of y on the sample point x is suppressed. The smaller the expected entropy is, the more certain we are about the whole space. If denoting the decision space as D, the sampled points as S and the rest of the space as $D - S$, the experimental design aims to find S such that the expected posterior entropy of $D - S$, i.e. $E[-\log p(D - S|S)]$ is minimized; or equally speaking, find S to minimize the overall uncertainty of the unexplored space. It has been shown that the problem phrased above is equivalent to maximizing the prior entropy of S (Sacks et al., 1989); and with the DACE model discussed above, it is further equivalent to finding design points that maximize $|R|$—the determinant of the correlation matrix for the sampling points. Maximizing $|R|$ is extremely difficult and there is no apparent method to solve it when the size of S is large. Nevertheless, if the size of S is restricted to 1, or when incremental sampling is performed, the optimal design point x is the one who has the maximum prediction variance $s^2(x)$ (or Bayesian information) and it can be obtained by maximizing (8). Because a decision point x furthest away from x_i has the largest prediction variance, maximizing Bayesian information implies global sampling and it tries to explore the space evenly.

2.4.2.2. Maximizing expected improvement. This method is developed under the deterministic global optimization context (Jones et al., 1998; Sasena et al., 2002). Let $f_m = \min(y_1, \ldots, y_n)$ denote the best performance value among all the n sampled points. For an arbitrary point x, consider the relation between its performance value $y(x)$ and f_m. According to (7) and (8), $y(x)$ follows a normal distribution $N(\tilde{y}(x), s^2(x))$, and the expected improvement (EI) is defined as:

$$I(x) \equiv E[\max(f_m - y(x), 0)] \tag{9}$$

This quantity actually reflects our belief (in Bayesian sense) that the new minimum would be $f_m - I(x)$ if x is sampled next. Clearly, the optimal x should maximize $I(x)$, and the following form of $I(x)$ is used in the optimization by analytically

carrying out the expectation operation on the right side of (9) (Jones et al., 1998):

$$I(x) = (f_m - \tilde{y}(x))\Phi\left(\frac{f_m - \tilde{y}(x)}{s(x)}\right) + s(x)\phi\left(\frac{f_m - \tilde{y}(x)}{s(x)}\right) \quad (10)$$

In Eq. (10), $\Phi(x)$ is the accumulative normal distribution function and $\phi(x)$ is the normal density function. Of the two terms in (10), the first term can be a large positive value near f_m, while the second term can be a large positive value when prediction variance $s^2(x)$ is large which occurs only when x is relatively far away from sampled points. Thus, the first term leads to local sampling while the second term leads to global sampling; however, the global sampling capability is rather weak.

2.5. Stopping criteria

Several criteria are implemented to control the termination of the loop in Fig. 1. The two most important criteria are absolute expected improvement defined by (9) and relative expected improvement defined by the ratio of the current expected improvement versus the initial expected improvement. These criteria will stop surrogate model building once they detect that further improvement drops below a prescribed value. In addition, total simulation time and total number of sampled points are also used as stopping criteria, the values of which can be set based on the computational budget.

3. Framework validation

We first validate the surrogate model framework before applying it to supply chain analysis. The purpose of the validation is to show that the surrogate model is able to capture the underlying input and output relations over the decision space of interest by only observing the performance values at sampled points. For this purpose, we choose the 'bra' function and 'cam' function (Dixon and Szegö, 1978) to return the performance values in place of the simulation; thus the input and output relation of the data is known. If the framework is effective, then the surrogate model it generates should adequately capture the structure of the 'bra' and 'cam' test function. That is, the surrogate model should indicate the neighbourhoods in which the optimal solutions are located.

Fig. 2 shows the contour profile of the 'bra' function, which has three local minimums in the domain of interest. For each sampling point x, a performance value bra(x) + N(0, 900) is returned, i.e. random noise following a normal distribution N(0, 900) is superimposed on the 'bra' function. The scaling is such as to cause the average noise to signal ratio to be as high as 0.3. Fig. 3 shows the contour profile of the resulting surrogate model. Note that the initial LHS uses 20 points and the final model uses 29 sampling points where each point incurs one simulation run. It is noteworthy that, even with relatively high level of noise presented

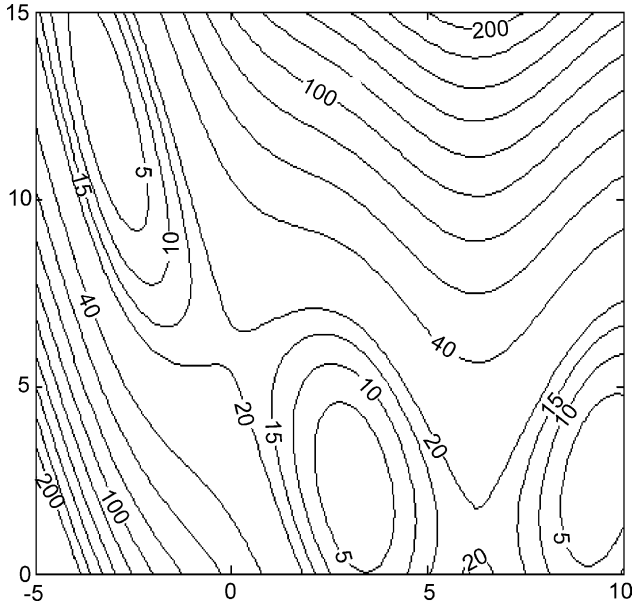

Fig. 2. Contour of 'bra' function.

in the data, the underlying structure of 'bra' function is still captured with the three local minimums clearly identified. Although the locations of the minima points of the surrogate do not agree exactly with those of the testing function, the results are absolutely acceptable since these minima points do correspond to good solutions (the optimal objective function value is no larger than 10 of the true optimum 0.4) of the original function (Fig. 2). Computational efficiency, the major strength of our proposed framework, is also manifested by this simple example. For purpose of comparison, we have applied the well-known algorithm SPSA (Spall, 1998) to the testing function. The results obtained show that even with smaller noise level, N(0, 100), and a local optimum as the

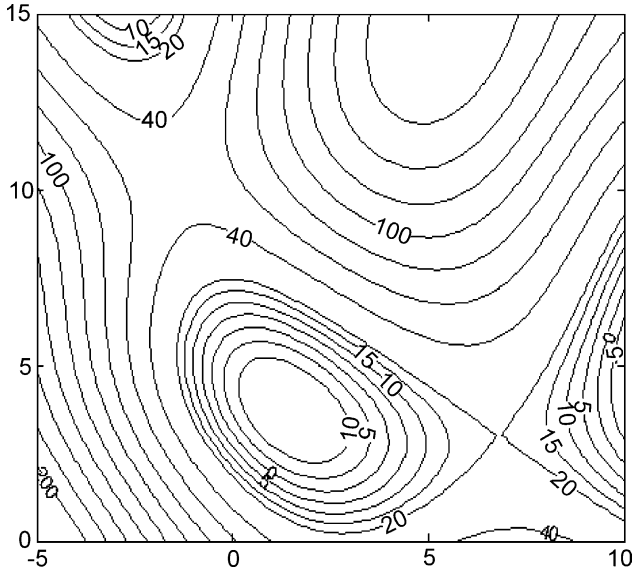

Fig. 3. Contour of the surrogate model for 'bra' function.

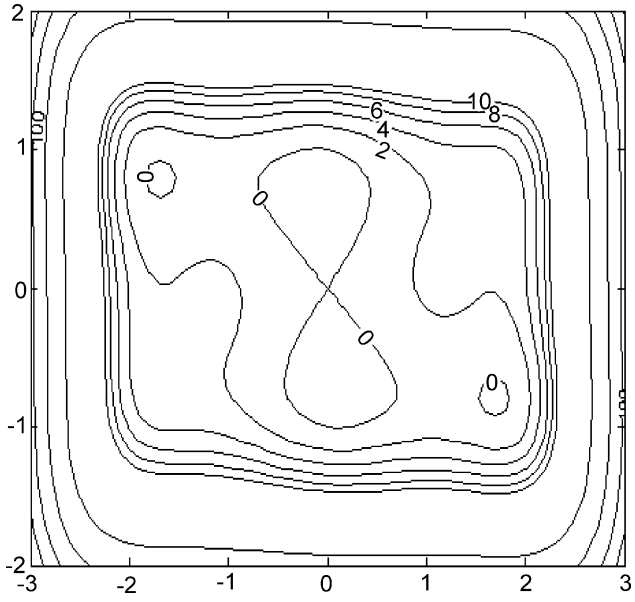

Fig. 4. Contour of 'cam' function.

initial point, 500 steps of search (where each step incurs two simulation runs) yield an objective function value 30 larger than the true optimum. Further increasing the search to 5000 steps does not improve the solution quality.

The contour profile of the 'cam' function is shown in Fig. 4, which indicates that the 'cam' function is much more complex than the 'bra' function; it has large values on the boundaries and most of the interior is rather flat with very fine structures. In a real application, this problem is trivial since most of the interior points are acceptable good solutions. However, the results in Fig. 5 demonstrate that the proposed framework can even capture the fine structures when normal distributed noise $N(0, 0.04)$ is superimposed. Initially, 40 points are provided by LHS; the final surrogate model only

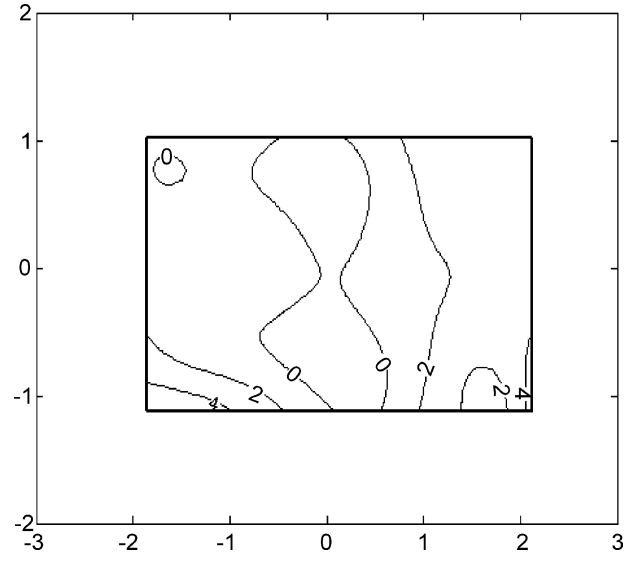

Fig. 5. Contour of the surrogate model for 'cam' function.

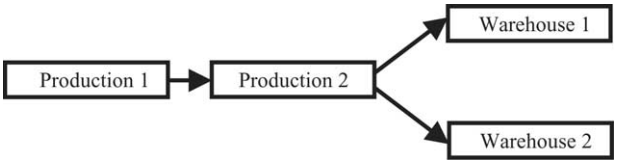

Fig. 6. The supply chain network structure.

describes the reduced domain represented by the black box in Fig. 5 due to the domain reduction and contains 39 points. Figs. 4 and 5 also illustrate the effectiveness of domain reduction and incremental sampling in terms of zooming in the interested regions in the domain.

4. Case studies

The proposed simulation-based optimization framework is next applied to a three-stage supply chain. The network structure is represented in Fig. 6. Products 1 and 2 are produced in the node production 1, and then shipped to the node production 2 where further processing is carried out. The finished goods from the node production 2 are transported to warehouses to meet customer demands. Warehouse 1 only stores product 1 and warehouse 2 only stores product 2.

The transportation times between nodes are random including the transportation from the outside supplier, who ships out all the demands on order and is not shown in Fig. 6, to the node production 1. All the transportation uncertainties are modelled with the triangular distribution Tri(5, 12, 8). The operation of each production node is modelled as a simple queuing process consisting of a single machine whose service times follow exponential distributions, and both products share the same machine. No additional implication for the optimization will occur if other more complex methods like MILP are used in the simulation to model the production; the only effect is that the simulation time will increase significantly as the simulation model is much more complex. The customer demands are random both in terms of the inter-arrival intervals and quantities. The unmet demands are fully backlogged by warehouses. The supply chain operates under a base stock policy. That is, the order cost is assumed to be 0 and each demand is transferred instantly to the upstream entity without any delay. Further, we assume that the chain operates under a pure "pull" mechanism, thus neither demand forecasting nor production planning are employed in this case study. The objective of the optimization is to minimize the backlogging costs at warehouses and the holding costs at all nodes by setting the ten base stock levels (each production node has inventories for raw materials and products).

The two products have quite different characteristics. Demand quantities for both products have the same triangular distribution Tri(4, 8, 6), but the demand arrival intervals for product 1 follow a Gamma distribution Gamma(12, 3), while that for product 2 follows a Gamma distribution Gamma(4, 3). Table 1 shows the holding costs and backlogging costs at

Table 1
Holding costs and backlogging costs

	h_{11}	h_{12}	h_{21}	h_{22}	h_{w1}	h_{w2}	b_{w1}	b_{w2}
Product 1	3	3	6	6	9	–	60	–
Product 2	1	1	2	2	–	3	–	20

Table 2
Average processing time

	t_{11}	t_{12}	t_{21}	t_{22}
Case 1	1	1	2.2	2.2
Case 2	1	1	2	2

the different locations of the supply chain. Note that h_{i1} ($i = 1$, 2) are the holding costs of raw materials at node production i, h_{i2} ($i = 1$, 2) are the holding costs of products at node production i, h_{wj} ($j = 1$, 2) are the holding costs at warehouse j, and b_{wj} ($j = 1$, 2) are the backlogging costs at warehouse j. Both products are produced in batch mode with a fixed batch size of 4. Table 2 lists the average processing time considered, where t_{ij} ($i = 1$, 2; $j = 1$, 2) represent average processing time of product i at production node j. Further, two production queuing modes are considered—mode 1 is simply first in first out (FIFO), while mode 2 gives priority to product 1, i.e. always produce product 1 if both products have demands in the queue.

4.1. The effect of operation policy and parameter

For this case problem, the performance value of each interested decision point was evaluated by one simulation run in the optimization process; every simulation run lasted 200,000 periods with the first 10,000 periods as the warm-up time. Table 3 shows a typical set of the optimization results returned by the framework. The subscripts in the headings follow the conventions introduced in Table 1; CIMJ ($I = 1$, 2; $J = 1$, 2) is the combination of average processing time in Table 2 with the two production modes; e.g. C1M1 indicates the production time of case 1 under the FIFO mode.

Table 3
Base stock policy

	I_{11}	I_{12}	I_{21}	I_{22}	I_{w1}	I_{w2}	Cost
C1M1							
Product 1	11.8	21.5	5.5	46.5	19.5	–	722
Product 2	26.4	77.6	25.4	75.3	–	73.8	
C1M2							
Product 1	23.6	22.1	2.5	11.7	9.0	–	603
Product 2	1.3	40.7	60.9	75.6	–	98.7	
C2M1							
Product 1	0	22.2	5.3	14.0	7.5	–	378
Product 2	0	18.1	24.5	51.3	–	44.8	
C2M2							
Product 1	0	8.3	0	0	16.2	–	282
Product 2	27.6	55.1	5.2	10.5	–	66.3	

Comparison of production mode 1 with production mode 2 shows that production mode 1 incurs a higher cost; the intuitive reason is that FIFO leads to the expensive product 1 having to undergo a longer production lead time by waiting in the queue for machine time, and a longer lead time inevitably causes a higher safety stock level. Table 3 also reveal that the average processing time has significant effect on the total cost, which is manifested by the much higher cost of case 1. These results imply that the scheduling/planning of the production units (they are represented as simple queuing systems in the case study) should be treated with more detail in studying the supply chain since it determines the processing time and lead time of each product and its impact on the supply chain optimization results can be significant. Though more details do not incur fundamental difficulty for simulation-based optimization, they do pose modelling and solving challenges for analytical methods, and that is where simulation-based optimization become a good option for supply chain optimization problems.

Figs. 7–9 demonstrate the progress of the optimization generally observed in the study, where the dash-dotted lines are the noisy simulation results of one simulation run per decision point used in the optimization and the solid lines represent the "true" average values of the corresponding decision points which are obtained by averaging 10 simulation runs per decision point. In those figures, each optimization trace starts from a positive number corresponding to the number of initial LHS, ends with a total number of simulated (or sampled) decision points that are different for different scenarios depending on when a stop criterion is reached; the difference between the total number of sampled decision point and the initial number of LHS is due to the incremental sampling of experimental design. Fig. 9 represents two scenarios with the upper two lines corresponding to C2M1 and the lower two lines corresponding to C2M2. The gaps between the solid lines and dash-dotted lines reflect the amount of noise contained in the simulation results. In addition to supporting the

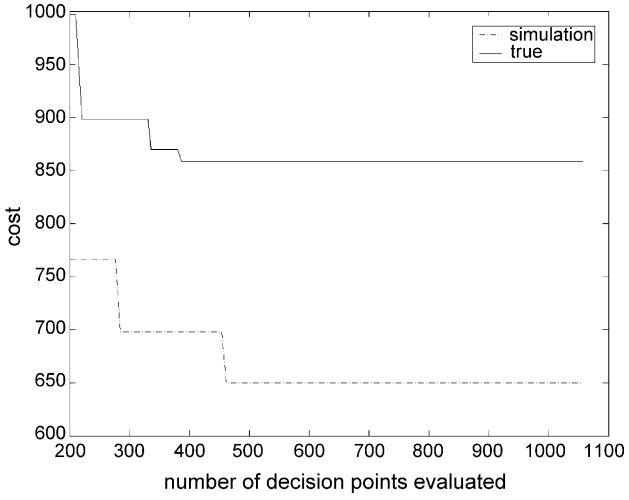

Fig. 7. An optimization trace for C1M1.

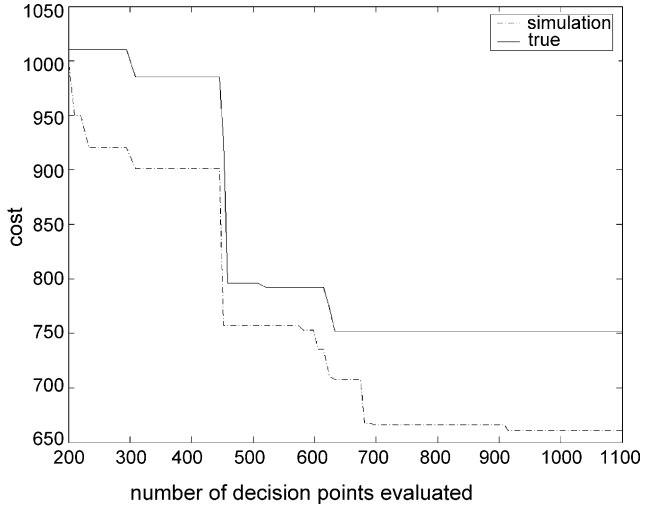

Fig. 8. An optimization trace for C1M2.

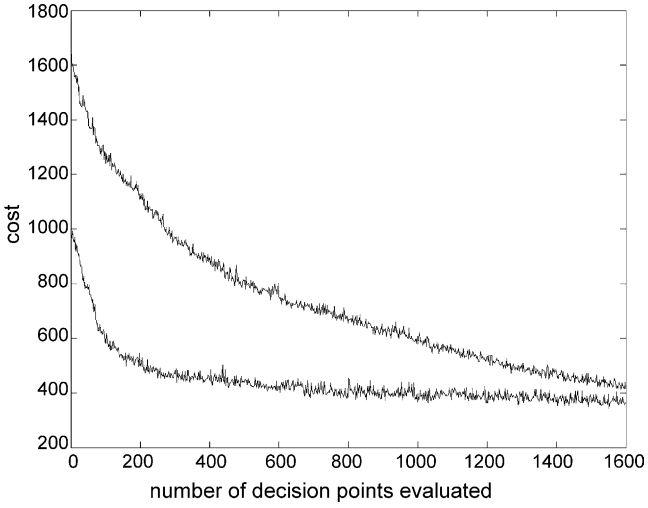

Fig. 10. Two SPSA traces for C2M2.

conclusions inferred from Table 2, these figures also indicate that the noise level in the data is greatly influenced by the operational policy and parameters.

4.2. Comparison with SPSA

The primary advantage of the proposed framework is to locate good solutions with better quality or with a smaller number of simulation runs (or equivalently sample a smaller total number of decision points as each decision point incurs one simulation run during optimization) in comparison with current available algorithms. To demonstrate this point, the case problem was solved with SPSA, one of the most efficient existing algorithms according to the simulation-based optimization literature (Spall, 1992, 1999). The optimization progress of SPSA is illustrated by Figs. 10 and 11 where only noisy simulation results based on one simulation run per decision point is shown without computing the "true" values as in Figs. 7–9. It is evident that for C2M2 (also holds for C2M1) SPSA can obtain solutions of similar quality with an approximately equal number of simulation runs as the proposed framework. However, for C1M2 (also for C1M1) SPSA can only identify inferior solutions even with a larger number of simulation runs. It is tempting to infer that SPSA may perform well when the noise is small based on the fact that C2M2 and C2M1 have lower noise level than C1M2 and C1M1. This is not true as will be clear later when scale-up issue is considered. In summary, the comparison manifests that the proposed framework is more effective than SPSA to optimize the performance of supply chains under uncertainty.

4.3. Chance constraints

Chance constraints are necessary to properly reflect decision makers' attitude in many situations. For example, de-

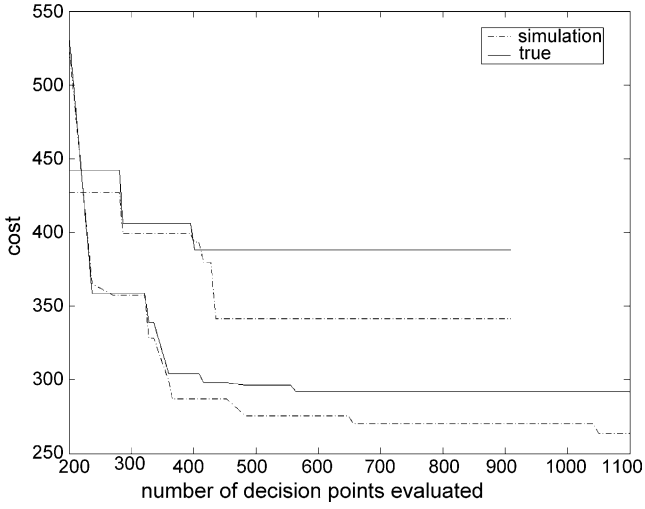

Fig. 9. Optimization traces for C2M1 (upper) and C2M2 (lower).

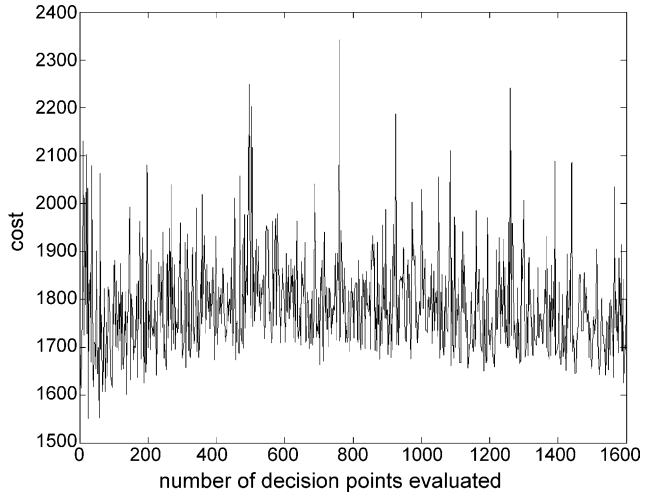

Fig. 11. A SPSA trace for C1M2.

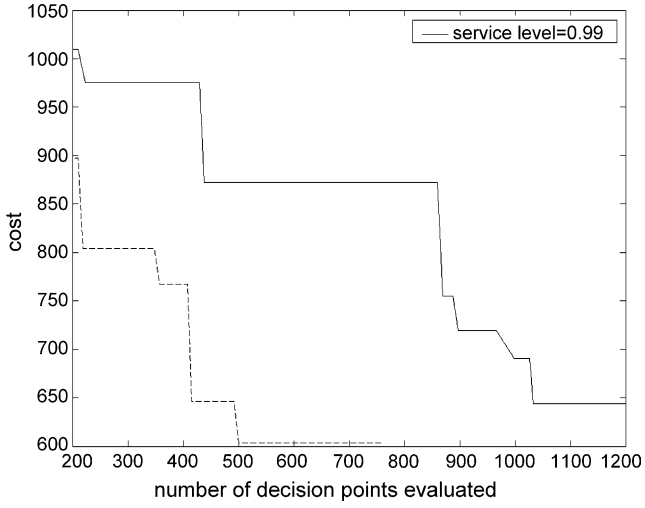

Fig. 12. Optimization traces for C1M2 with and without service level requirement.

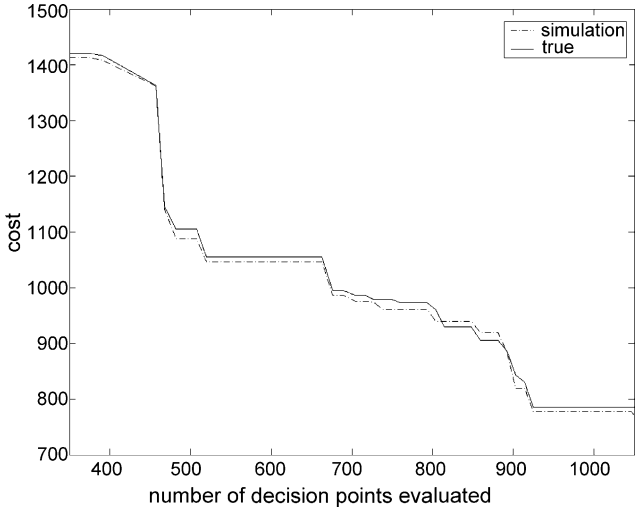

Fig. 14. An optimization trace for the case problem 2.

cision makers may require certain service levels to be met in addition to penalizing the backlogging while optimizing safety stocks. The proposed surrogate model framework can handle chance constraints with a simple idea: inflate the cost to an artificially large value once the simulation detects that a decision point violates the chance constraints, i.e. the decision point is infeasible. Because the costs of the infeasible decision points are so high, the domain reduction procedure will cut off those points from consideration. Even occasionally some of those points are not excluded from the interested domain, the optimization with the surrogate model will not mistakenly treat those points as good solutions because they have high costs. Though it might be possible to modify the existing algorithms like SPSA and RSM to accommodate chance constraints, it is not easy to achieve the goal without procedures like domain reduction. From this perspective, the proposed framework is more suitable for problems involving chance constraints. Figs. 12 and 13 show the results of C1M2 and C2M2 with and without service level requirement. It is true that imposing service level equal to 0.99 will lead to cost no smaller than that of without the service level constraint under the condition that optimal decisions are considered for both situations. However, since the proposed framework, similar to all other algorithms, does not guarantee to locate optimal decisions, there exist instances where the costs with service level constraints are actually smaller than the costs without service level constraints, as illustrated by Fig. 13.

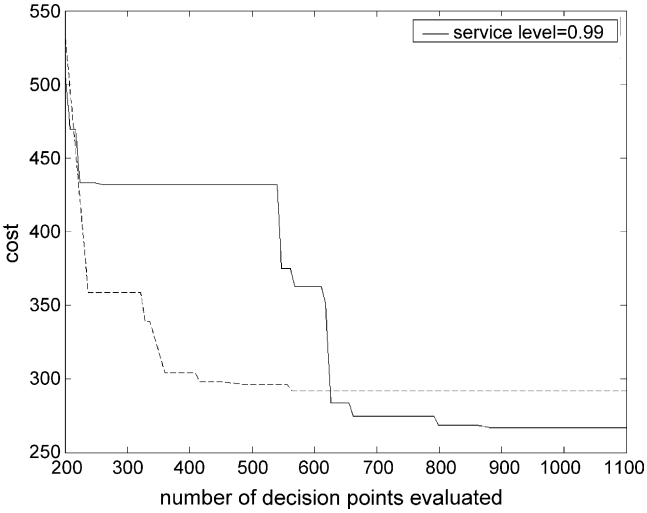

Fig. 13. Optimization traces for C2M2 with and without service level requirement.

4.4. Scale-up issues

One potential drawback of the surrogate model approach is the scale-up issue. As the dimensionality of the decision space goes up, the number of decision points needed to construct the surrogate model may increase exponentially. One of

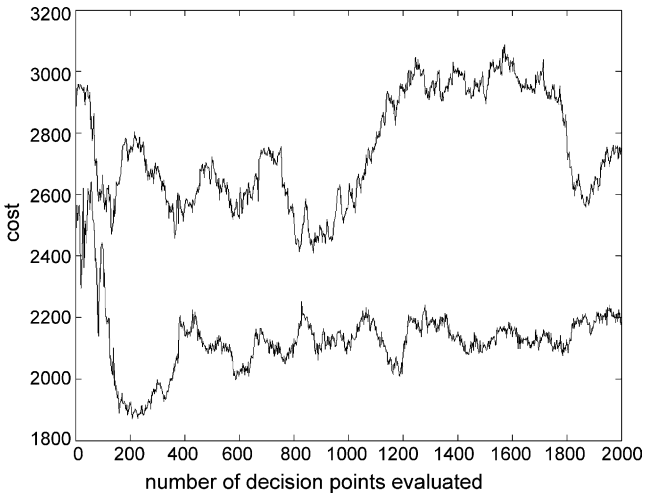

Fig. 15. Two SPSA traces for the case problem 2.

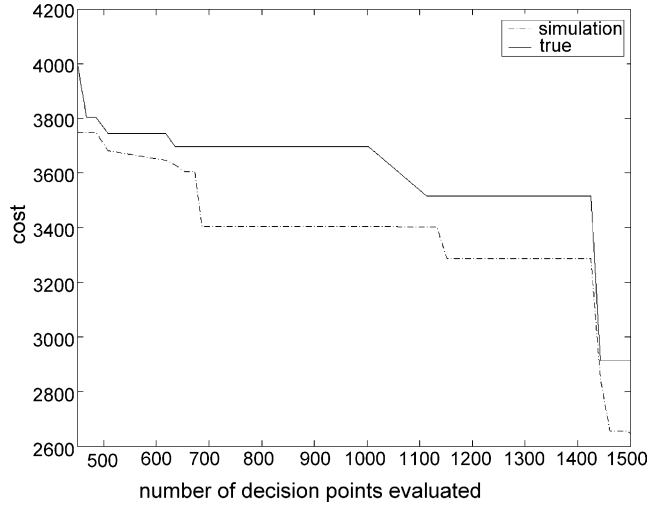

Fig. 16. An optimization trace for the case problem 3.

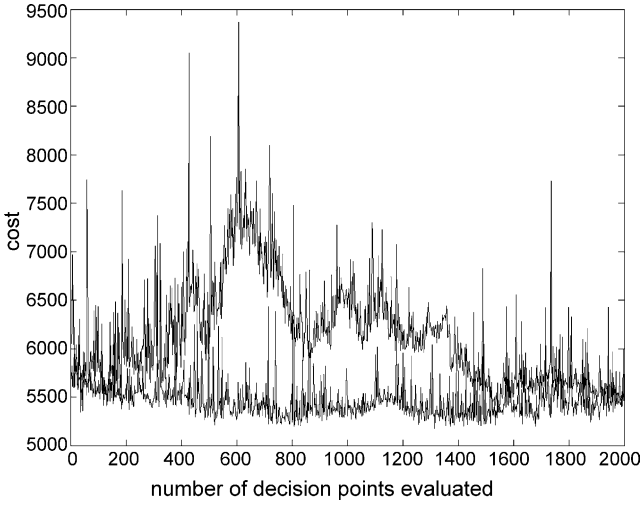

Fig. 17. Two SPSA traces for the case problem 3.

the reasons that RSM is not successful for high-dimensional problems is that they need $O(n^2)$ number of points to fit a full factor second-order surface. In this aspect, SPSA exhibits an excellent property; it always computes a gradient with two decision points (or simulation runs) independent of the dimensionality of the decision space.

The case problem studied above only contains 10 decision variables. To examine whether the proposed framework scales well, two more case problems, case problem 2 and case problem 3 were constructed with 20 and 40 decision variables, respectively, by increasing the number of products from 2 to 4 and 8. Basically, the original two products were replicated except for the production time. For case problem 2, all four products have average processing time 0.8 at both production nodes. For case problem 3, the first two products have average processing 0.5 at both production nodes, and the rest six products have average processing time 0.4 at both production nodes. To account for the larger number of random variables, the simulation horizon for the two new case problems are increased to 300,000 and 400,000, respectively. The results of the proposed framework are again compared with those of SPSA. Figs. 14–17 clearly show that for the higher-dimensional problems, SPSA performs badly even when there is little noise in the simulation results as manifested by Figs. 14 and 15. Thus, the dimensionality independence property of SPSA does not actually turn into the desired result of good solutions with a small number of

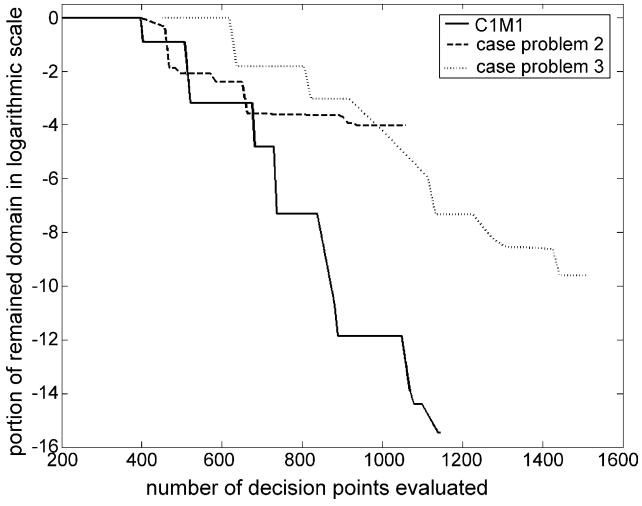

Fig. 18. Domain reduction traces.

simulation runs. These results somehow indicate that the proposed surrogate framework may not experience the suspected scale-up problems as the number of sampled points does not dramatically increase. A possible explanation for this phenomenon is that domain reduction will restrict the surrogate model to promising regions where good solutions might exist and the important structures of these regions do not necessarily need a large number of samples to expose; Fig. 18 shows that due to the reduction, the final domains in general were

Table 4
Average costs of different scenarios

	Policy					
	FIFO				Priority	
	10[a] (C1)	10[a] (C2)	20[a]	40[a]	10[a] (C1)	10[a] (C2)
Surrogate model	890	390	800	2950	650	310
SPSA	1900	420	2400	5500	1700	370
No reduction and design	940	430	1380	3410	810	420

[a] Number of decision variables.

only a small fraction of the initial domains, and their volume ratios can be as small as 1.0E−16.

Table 4 summarizes the results for all case problems with different number of decision variables and production policies. Those results demonstrate that the proposed framework significantly out-performed SPSA in most of the scenarios and never under-performed. In comparison with the classical surrogate model algorithm where no domain reduction and adaptive experimental design are conducted, the proposed framework also located much better decisions especially for the case problems with 20 and 40 decision variables. To insure that both algorithms examine roughly equal number of decision points, 700, 1000 and 1500 points were sampled in the classical surrogate model algorithm for 10-, 20- and 40-dimensional problems, respectively.

5. Conclusions

An extension to the simulation-based optimization framework is proposed for optimizing supply chains under uncertainty. This framework employs a surrogate model together with domain reduction and incremental sampling to extract structure information from noisy simulation results; the supply chain decisions can then be efficiently optimized using this surrogate model. This framework is shown to correctly capture the structural features of two test functions superposed with random noises and to use simulation runs parsimoniously. Application to optimize the base stocks for a three-stage supply chain further demonstrates the power of the method. Comparison with SPSA algorithm illustrates that the proposed framework can generally obtain better solutions with a smaller number of simulation runs. The framework can also readily handle chance constraints and does not present serious scale-up problems.

References

Barton, R. R. (1994). Metamodeling: a state of the art review. In J. D. Tew, S. Manivannan, D. A. Sadowski, & A. F. Seila (Eds.), *Proceedings of the 1994 Winter Simulation Conference* (p. 237).

Chambers, M., & Mount-Campbell, C. A. (2002). Process optimization via neural network metamodeling. *International Journal of Production Economics*, *79*, 93.

Fu, M. C. (2002). Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing*, *14*, 192.

Greenwood, A. G., Paul Rees, L., & Siochi, F. C. (1998). An investigation of the behavior of simulation response surface. *European Journal of Operational Research*, *110*, 282.

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, *13*, 455.

Jung, J. Y., Blau, G., Pekny, J. F., Reklaitis, G. V., & Eversdyk, D. (in press). A simulation based optimization approaches to supply chain management under demand uncertainty. *Computers and Chemical Engineering*.

Kleinman, N. L., Spall, J. C., & Naiman, D. Q. (1999). Simulation-based optimization with stochastic approximation using common random numbers. *Management Science*, *45*, 1570.

Mackay, D. J. C. (1992). Bayesian interpolation. *Neural Computation*, *4*, 415.

Mackay, D. J. C. (1995). Probable networks and plausible predictions—A review of practical Bayesian methods for supervised neural networks. *Computation in Neural Systems*, *6*, 496.

Neddermeijer, H. G., van Oortmarssen, G. J., Piersma, N., & Dekker, R. (2000). A framework for response surface methodology for simulation optimization. In *Proceedings of the 1999 Winter Simulation Conference*.

Owen, A. B. (1998). Latin supercube sampling for very-high dimensional simulations. *ACM Transactions on Modeling and Computer Simulation*, *8*, 71.

Sabuncuoglu, I., & Touhami, S. (2002). Simulation metamodelling with neural networks: An experimental investigation. *International Journal of Production Research*, *40*, 2483.

Sacks, J., Welch, W. J., Michell, T. J., & Wynn, H. P. (1989). Design and analysis of computer experiments (with discussion). *Statistical Science*, *4*, 409.

Sasena, M. J., Papalambros, P., & Goovaerts, P. (2002). Exploration of metamodeling sampling criteria for constrained global optimization. *Engineer Optimization*, *34*, 263.

Schölkof, B., & Smola, A. J. (2002). *Learning with kernels*. Cambridge, MA: MIT Press.

Spall, J. C. (1992). Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, *37*, 332.

Spall, J. C. (1998). Implementation of the simultaneous perturbation algorithm for stochastic optimization. *IEEE Transactions on Aerospace and Electronic Systems*, *34*, 817.

Spall, J. C. (1999). Stochastic optimization and the simultaneous perturbation method. In *Proceedings of the 1999 Winter Simulation Conference*.

Subramanian, D., Pekny, J. F., & Reklaitis, G. V. (2000). A simulation-optimization framework for addressing combinatorial and stochastic aspects of R&D pipeline management problem. *Computers and Chemical Engineering*, *24*, 1005.

Subramanian, V., Wan, X., Pekny, J. F., & Reklaitis, G. V. (2003). Computational framework for supply chain analysis. *AIDIC Conference Series*.

Van Gestel, T., Suykens, J. A. K., Baestaens, D., Lambrechts, A., Lanckriet, G., & Vandaele, B. (2002). Financial time series prediction using least squares support vector machines within the evidence framework. *IEEE Transactions on Neural Networks*, *12*, 808.

Vapnik, V. N. (1998). *Statistical learning theory*. New York: Wiley.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, *10*, 988.