Surrogate-Based Agents for Constrained Optimization

Diane Villanueva[1,2*], Rodolphe Le Riche[2,3†], Gauthier Picard[2‡], and Raphael T. Haftka[1§]

[1] *University of Florida, Gainesville, FL, 32611, USA*

[2] *École Nationale Superiéure des Mines de Saint-Étienne, Saint-Étienne, France*

[3] *CNRS UMR 5146*

Multi-agent systems have been used to solve complex problems by decomposing them into autonomous subtasks. Drawing inspiration from both multi-surrogate and multi-agent techniques, we define in this article optimization subtasks that employ different approximations of the data in subregions through the choice of surrogate, which creates surrogate-based agents. We explore a method of design space partitioning that assigns agents to subregions of the design space, which drives the agents to locate optima through a mixture of optimization and exploration in the subregions. These methods are illustrated on two constrained optimization problems, one with uncertainty and another with small, disconnected feasible regions. It is observed that using a system of surrogate-based optimization agents is more effective at locating the optimum compared to optimization with a single surrogate over the entire design space.

Nomenclature

c	=	centroid
f	=	objective function
$\mathbb{F}$	=	objective function values associated with design of experiments in database
g	=	constraint
$\mathbb{G}$	=	constraint values associated with design of experiments in database
t	=	time
x	=	design variables
$\mathbb{X}$	=	design of experiments in database

I. Introduction

Complex optimization problems often require repeated computationally expensive simulations, such as those found in high fidelity finite element analyses. Therefore, many researchers in the field have focused on the development of optimization methods adapted to expensive simulations. The main ideas underlying such methods are i) the use of surrogates ii) problem decomposition iii) problem approximation iv) parallel computation.

In computer science, multi-agent systems solve complex problems by decomposing them into autonomous sub-tasks. A general definition posits a multi-agent system to be comprised of several autonomous agents with possibly different objectives that work toward a common task. In terms of optimization, a multi-agent system could solve decomposed problems such that the agents only know subproblems.

*Graduate Research Assistant, Mechanical and Aerospace Engineering/Institut Henri Fayol, AIAA student member

†CNRS permanent research associate, Institut Henri Fayol

‡Associate Professor, Institut Henri Fayol

§Distinguished Professor, Mechanical and Aerospace Engineering, AIAA Fellow

In a past study,[1] we described a methodology to approximate complex, computationally expensive reliability-based design optimization problems, and use the approximated problems to efficiently find a solution with a high level of accuracy. The process took advantage of multi-agent techniques. Each agent built a different surrogate and used it to find an optimum. In effect, this process creates "surrogate-based optimization agents". Each agent defined a different search process. Through cooperation (exchange of search points), the agents system defined high-level methodologies whose efficiency and robustness were studied.

The use of surrogates to replace expensive simulations and experiments in optimization has been well documented.[2–5] To take advantage of parallel computing, many have proposed strategies for using multiple surrogates for optimization.[6–9] Surrogate-based agents are related to multi-surrogates, but the reference to "agents" stresses that agents can change strategies at runtime and their actions are asynchronous. There are several strategies that agents may employ:

1. Agents solve the same problem with different strategies.[1]

2. Agents solve parts of the complete problem. A typical example is given by multi-criteria optimization, where the criteria would be split among agents as seen in approaches based on game theory. Another example is to partition the design space such that each agent has to search in a smaller subregion.

In all cases, the agents' individual strategies are completed by a coordination mechanism, which, in the context of surrogate-based optimization, will consist of a policy for sharing the results of high fidelity simulations. In addition, agents can be deleted if they are not found to be useful and, vice versa, new agents can be created.

In our current study, the partitioning of a design space to define an agent's subregion is of particular interest. There are several existing methods of fitting local surrogates to subregions of the design space that are partitioned using clustering techniques. Zhao and Xue[10] presented a methodology to divide the design space into clusters using the Gaussian mixture model. The Gaussian mixture model assigned points to clusters by maximizing membership probabilities. Local surrogates were fit to each cluster, and a global surrogate was fit to the entire space, merging the local surrogates by Bayes' law. Wang and Simpson[11] used response surfaces, kriging models, and points sampled from these surrogates to reduce the design space to only interesting regions. The surrogates were used to generate inexpensive points, which were then clustered using fuzzy c-means clustering. The purpose of the clustering was to identify a region with low-function values for further high-accuracy approximation and discard regions with high function values.

The majority of the aforementioned research focused on unconstrained optimization problems. In constrained surrogate-based optimization, Sasena et al.[12]extended the efficient global optimization (EGO)[13] algorithm such that the sampling criterion considered the probability of feasibility of a point rather than the expected improvement on the present best solution. The method was shown to be efficient at solving problems where feasible regions of the design space were small or disconnected. Other approaches to solving constrained optimization problems include a two-phase process in which the first phase finds a feasible solution and the second phase solves a bi-objective problem where the original objective function and measure of constraint violation were the objectives. This was a method put into place in genetic algorithms by Venkatraman and Yen.[14] This was a development in an approach to constraint handling by a bi-objective formulation used earlier by Fletcher and Leyffer,[15] that has also been applied to genetic algorithms and to particle swarm optimization.[16–18]

In this article, we provide a background on surrogate-based agents. We then describe the methods of design space partitioning and simulation point allocation that are intended to distribute local optima among the partitions while maximizing the accuracy of the surrogate. Finally, these methods are illustrated in two example problems: a one dimensional system-level reliability optimization problem and a two dimensional optimization problem with a single constraint and disconnected feasible regions.

II. Definition of Agents for Surrogate-Based Optimization

Let us consider the general formulation of a constrained optimization problem shown in Eq.(1).

$$\begin{aligned} \underset{x\in\mathcal{S}\subset\Re^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g(x)\leq 0 \end{aligned} \tag{1}$$

In surrogate-based optimization, a surrogate is built from a design of experiments (DOE), denoted by $\mathbb{X}$ that consists of sets of the design variables x. For the design of experiments, there are the calculated values of the objective function f and constraints g that are associated with the DOE, which we denote as $\mathbb{F}$ and $\mathbb{G}$, respectively. We will refer to $\mathbb{X}$ and its associated values of $\mathbb{F}$ and $\mathbb{G}$ as a database.

The database is used to construct the surrogate approximation of the objective function $\hat{f}$ and the approximation of the constraint $\hat{g}$. We can approximate the problem in Eq.(1) using the surrogates and formulate the problem as

$$\begin{aligned} \underset{x \in \mathcal{S} \subset \Re^n}{\text{minimize}} \quad & \hat{f}(x) \\ \text{subject to} \quad & \hat{g}(x) \leq 0 \end{aligned} \tag{2}$$

The solution of this approximate problem is denoted $\hat{x}^*$.

Surrogate-based optimization calls for more iterations to find the true optimum, and is therefore dependent on some iteration time t. That is, after the optimum of the problem in Eq.(2) is found, the true values of f and g are calculated and included in the DOE along with $\hat{x}^*$. In the next iteration, the surrogate is updated, and the optimization is performed again. Therefore, we denote the DOE at a time t as $\mathbb{X}^t$ and the associated set of objective function values and constraint values as $\mathbb{F}^t$ and $\mathbb{G}^t$, respectively. The surrogate-based optimization procedure is summarized in Algorithm 1. Note that in the algorithm the

Algorithm 1 Procedure for surrogate-based design optimization

```
1: t = 1 (initial state)
2: while $t \leq t^{max}$ do
3:     Build surrogates $\hat{f}$ and $\hat{g}$ from $(\mathbb{X}^t, \mathbb{F}^t, \mathbb{G}^t)$
4:     Global optimization $\rightarrow \hat{x}^*$ (Here, solve: $\underset{x \in \mathcal{S}}{\text{minimize}}\ \hat{f}(x)$ subject to $\hat{g}(x) \leq 0 \rightarrow \hat{x}^*$
5:             if $\hat{x}^*$ is near $\mathbb{X}^t$ or infeasible, solve $\underset{x \in \mathcal{S}}{\text{maximize}}$ distance$(\mathbb{X}^t) \rightarrow \hat{x}^*$)
6:     Calculate $f(\hat{x}^*)$ and $g(\hat{x}^*)$
7:     Update database $(\mathbb{X}^{t+1}, \mathbb{F}^{t+1}, \mathbb{G}^{t+1}) \cup (\hat{x}^*, f(\hat{x}^*), g(\hat{x}^*))$
8:     $t = t + 1$
9: end while
```

alternative surrogate optimization and space filling together define an infill sampling criterion to find $\hat{x}^*$. In global kriging-based optimization, the infill sampling criterion can be based on, for example, expected improvement or probability of improvement on the present best solution. Such criteria consider the uncertainty structure of the kriging surrogate, along with the prediction provided by the surrogate to explore the design space. In this paper, the surrogate is used to solve the actual optimization problem shown in Eq.(2) using only the surrogate prediction. This strategy would likely be ineffective with a single surrogate over the design space as it could lead to convergence at a single point that is possibly away from the true optimum, which is the reason why it needs to be completed with space filling phases.

Algorithm 1 can be thought of as the procedure followed by a single surrogate-based agent in one iteration. In addition, when there is more than one agent, each agent is restricted to only a subregion of the design space, i.e., $\mathcal{S}$ is replaced by a part of $\mathcal{S}$. The rationale behind this idea is that each agent has an easier optimization subproblem to solve because it searches a smaller space, which we denote as $\mathcal{P}_i$ for the ith agent. Each agent must consider only the points in its subregion, which are available in its internal database $(\mathbb{X}^t, \mathbb{F}^t, \mathbb{G}^t)_{internal}$. The subregion of an agent is defined by the position of its centroid c. A point in the design space belongs to the subregion with the nearest centroid, where the distance is the Euclidean distance. This creates subregions that are Voronoi cells.[19] The choice of where to place the centroid is the subject of the next section.

Figure 1 illustrates the partition of a two-dimensional design space into four subregions for four agents, which requires four centroids. In this example, we place the centroids randomly. The points in the design space that correspond to each subregion are represented by the four different colors.

Once the subregions are defined, each agent fits several surrogates and chooses the one that maximizes the accuracy in its subregion. To avoid ill-conditioning, if more points are needed than are available to an agent, the agent asks neighboring agents for points. The neighboring agents then communicate the information associated with these points. We define the best surrogate as the one with the minimum cross-validation error, the partial prediction error sum of squares $PRESS_{RMS}$. This is found by refitting the surrogate and

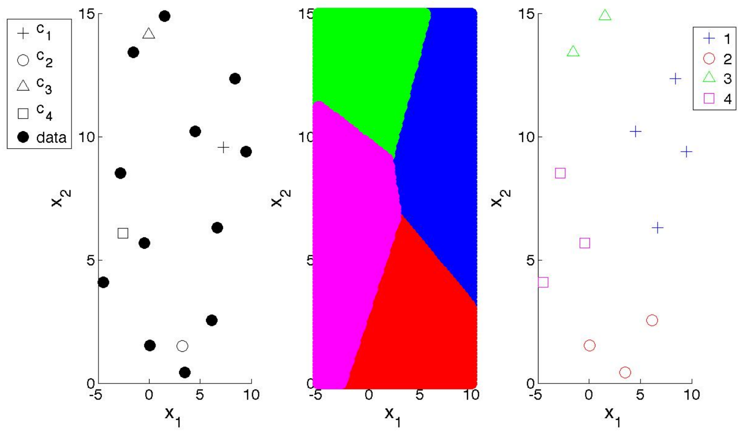


Figure 1. Two-dimensional partitioning example showing data points and four centroids (left), corresponding to four agents and subregions (middle), and assignment of data points to each agent and subregion (right).

leaving out a point, and measuring the error at that point. This is done for p points in the agent's subregion (disregarding any points received from other agents) to form a vector of the cross-validation errors e_{XV}. The value of $PRESS_{RMS}$ is then calculated by

$$PRESS_{RMS} = \sqrt{\frac{1}{p} e_{XV}^T e_{XV}} \tag{3}$$

Once the agents have chosen surrogates, the optimization is performed to solve the problem in Eq.(2) inside the subregion. If the optimizer gives an infeasible point (i.e., the point does not satisfy the constraint in Eq.(2) or is out of the subregion) or repeats an existing point, the agent then explores in the subregion. To explore, the agent adds a point to the database that maximizes the minimum distance from the points already in its internal database. The true values f and g of the iterate are then calculated, and $(\hat{x}^*, f(\hat{x}^*), g(\hat{x}^*))$ is added to the global database. The next iteration then starts by repartitioning the design space using the points in the global database. The procedure of a single agent is given in Algorithm 2.

Algorithm 2 Agent i optimization in its subregion. The method of partitioning is described in Algorithm 3

```
 1: t = 1 (initial state)
 2: while $t \leq t^{max}$ do
 3:     Partition the design space (Algorithm 3), get $c_i$
 4:     Define the local design space as $\mathcal{P}_i = \{x \in \mathcal{S} \text{ s.t.} ||x - c_i||^2 \leq ||x - c_j||^2 \ , \ j \neq i\}$
 5:     Update internal database from the new design space partition
 6:     Build surrogates $\hat{f}$ and $\hat{g}$ from internal database $(\mathbb{X}^t, \mathbb{F}^t, \mathbb{G}^t)_{internal}$
 7:     if Not sufficient number of points in internal database to build a surrogate then
 8:         Get points from other agents closest to $c_i$
 9:         Build surrogates
10:     end if
11:     Choose best surrogate based on partial $PRESS_{RMS}$ error
12:     Global optimization $\rightarrow \hat{x}^*$ (Here, solve: $\underset{x \in \mathcal{P}_i}{\text{minimize}}\ \hat{f}(x)$ subject to $\hat{g}(x) \leq 0 \rightarrow \hat{x}^*$
13:             if $\hat{x}^*$ is near $\mathbb{X}^t$ or infeasible, solve $\underset{x \in \mathcal{P}_i}{\text{maximize}}$ distance$(\mathbb{X}^t) \rightarrow \hat{x}^*$)
14:     Calculate $f(\hat{x}^*)$ and $g(\hat{x}^*)$
15:     Update global database $(\mathbb{X}^{t+1}, \mathbb{F}^{t+1}, \mathbb{G}^{t+1})_{global} \cup (\hat{x}^*, f(\hat{x}^*), g(\hat{x}^*))$
16:     $t = t + 1$
17: end while
```

III. Design Space Partitioning

The method of design space partitioning we propose focuses on moving the subregions' centroids to different local optima. As a result, each agent can choose a surrogate that is accurate around the local optimum, and the agent can also explore the subregion around the local optimum.

However, for the initial partitioning, in which there is no information on local optima, we follow a different logic: the initial partition is made by k-means clustering.[20] K-means clustering is a method that partitions n data points into k clusters such that each observation belongs to the cluster with the nearest mean. After the points are clustered in the first iteration ($t = 1$), agents are assigned to the partitioned subspaces and the points in the subspaces define the internal database $(\mathbb{X}^t, \mathbb{F}^t, \mathbb{G}^t)_{internal}$ of each agent. The process then follows the algorithm in Algorithm 2 until the end of the first iteration.

From the second iteration on, the centroid of the subregion is the database point in the subregion at $t-1$ that satisfies the constraint with the minimum value of the objective function. If none of the subregion points satisfy the constraint, then the centroid is the subregion point with the smallest constraint violation.

The placement of the centroid is determined by comparing the optimum of the last iteration $\hat{x}^{*t-1}$ against the centroid of the last iteration c^{t-1}. The location of the centroid is moved to the optimum of the last iteration if it improves (minimizes) the value of the objective function and if both $\hat{x}^{*t-1}$ and c^{t-1} satisfy the constraint. It is also moved when the last optimum satisfies the constraint and the last centroid does not. The centroid also moves when both the last optimum and last centroid do not satisfy the constraint and the last optimum provides a value of g that is closer to the feasible state. Finally, if the centroid is within a certain distance ε from another centroid, then the centroid is moved to maximize the minimum distance between the centroids. This moves the agent to a previously unexplored region of the design space. These different conditions are summarized in Algorithm 3, where after partitioning, the solution process continues with Algorithm 2.

Algorithm 3 Partitioning design space (calculate c_i's) around local optimum in subregion i. The iteration is represented by t.

```
1:  if $t = 1$ (initial state) then
2:      Use k-means clustering to find centroids to partition design space $\rightarrow c_i^t$
3:  else
4:      if $g(c_i^{t-1}) \leq 0$ then
5:          if $g(\hat{x}_i^{*t-1}) \leq 0$ & $f(\hat{x}_i^{*t-1}) < f(c_i^{t-1})$ then
6:              $c_i^t = \hat{x}_i^{*t-1}$
7:          else
8:              $c_i^t = c_i^{t-1}$
9:          end if
10:     else
11:         if $g(\hat{x}_i^{*t-1}) < g(c_i^{t-1})$ then
12:             $c_i^t = \hat{x}_i^{*t-1}$
13:         else
14:             $c_i^t = c_i^{t-1}$
15:         end if
16:     end if
17: end if
18: if $\parallel c_i^t - c_{j\neq i}^t \parallel \leq \varepsilon$ then
19:     Solve: $\max_{c_i \in \mathcal{S}} min(\parallel c_i^t - c_{j\neq i}^t \parallel) \rightarrow c_i^t$ {Delete current agent and create exploratory agent}
20: end if
```

IV. Illustrative Examples

In this section, two examples are provided to illustrate the effectiveness of the methods described in Sec. II and III. The first example is a one dimensional system-level reliability based optimization problem. The second example is a two dimensional optimization problem with disconnected feasible regions that cover approximately 3% of the design space.

In each example problem, the agents were given a set of surrogates from which they found the best surrogate based on $PRESS_{RMS}$. The set of surrogates includes response surfaces, kriging, and moving least squares. The set of surrogates and the minimum number of points used to fit each surrogate are provided in Table 1. The local optimization problems were solved with a sequential quadratic programming (SQP) algorithm.[21]

Table 1. Surrogates considered in this study

ID	Description	# of pts for fit
1	Linear response surface	
2	Quadratic response surface	1.5 * # of coefficients
3	Cubic response surface	
4	Kriging (quadratic trend)	
5	Kriging (linear trend)	1.5 * # coefficients for quadratic response surface
6	Kriging (constant trend)	
7	Moving least squares (Linear Shepard)	Entire DOE

A. System-Level Reliability-Based Optimization

1. *Problem Description*

The one dimensional example problem shown in Eq.(4) considered the probability of failure of a series system. In a series system, the system fails if there is failure in any single mode of failure.

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & f(x) = x sin\left(2x - \frac{1}{3}\right) + \frac{1}{2}x^2 + 0.7x + 0.2 \\
\text{subject to} \quad & p_{f,sys} = P(g_1(x,u_1) > 0 | g_2(x,u_2) > 0) \leq 0.1 \\
& -3 \leq x \leq 3 \\
\text{where} \quad & g_1(x,u_1) = \frac{1}{50}(20x cos(5x) + x^3 + x^2) + u_1(x) \\
& g_2(x,u_2) = -1.55 + e^{0.8x-0.4} + u_2(x) \\
& u_1 \sim N\left(0, \left(\frac{x^2+1}{10}\right)^2\right) \\
& u_2 \sim N(0, (0.2x^2)^2)
\end{aligned}
\tag{4}
$$

There are two failure modes, described by the constraints g_1 and g_2, which are functions of the design variable x and some uncertainty in the modeling of the failure mode, represented by u_1 and u_2. Both are normally distributed with a mean of zero and a variance that is a function of x. The optimum is located at $x^* = -1.43$ where $f(x^*) = 0.15$ and $p_{f,sys} = 0.10$.

In this example, both the objective function f and system probability of failure $p_{f,sys}$ were considered to be expensive to calculate, so both were approximated with surrogates. Since there is only uncertainty on the constraints and not on the design variable, it is simple to analytically calculate the system probability of failure (by considering g_1 and g_2 as a gaussian vector, then transforming it into the standard normal space). We assume that the underlying expense in calculating $p_{f,sys}$ comes from evaluating the nominal values of g_1 and g_2. Also, the surrogates of the system-level reliability index were constructed in place of the system-level probability of failure. The two are related by $p_{f,sys} = \Phi(-\beta_{sys})$, where Φ is the standard normal cumulative density function. For a system probability of failure of 0.1, the reliability index β_{sys} is 1.28. The values of f, $p_{f,sys}$, β_{sys}, g_1, and g_2 over the design space are shown in Fig. 2.

The seven possible surrogates that were available to each agent are described in Table 1. Three cases were compared: 3 agents with initial DOE of size 5 for $t^{max} = 10$ iterations, 1 agent with initial DOE of size 5 for $t^{max} = 30$ iterations, and 1 agent with initial DOE of size 20 for $t^{max} = 15$ iterations. This allowed

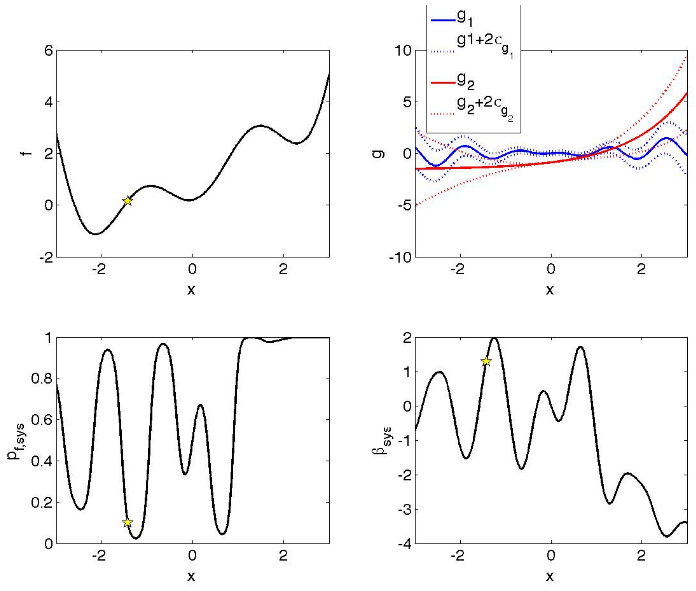


Figure 2. Plots of f (top left), g_1 and g_2 (top right), $p_{f,sys}$ (bottom left), and β_{sys} over x. The location of the optimum at -1.43 is represented by the yellow star.

for an equal number of calls to calculate f and $p_{f,sys}$ (35) in each case. The initial DOEs were found using Latin Hypercube sampling.

2. Results

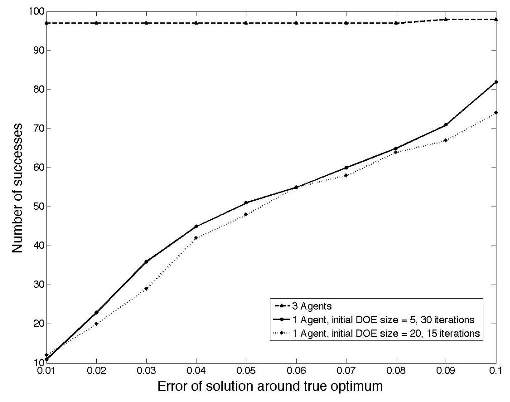
The three agent system was able to locate a feasible solution in all of the 100 repetitions, as was the single agent with the initial DOE of size 5 with 30 iterations. The single agent with an initial DOE of size 20 with 15 iterations was able to locate a feasible solution in 88 of the repetitions. Figure 3(a) compares the number of successes in 100 repetitions in which the three agent system and single agent were able to locate a solution with some error around the true optimum of $x = -1.43$. For the smallest error of 1% around the optimum, the three agent system was successful 97 times. This 1% error is an error of approximately 0.014 in x. When this error was increased to 8%, which is an error of approximately 0.11 in x, the agents were successful in locating a solution 98 times. The system of three agents was much better at locating a solution compared to both cases with single agent, which was only successful at most 12 times at 1% error and 82 times at 10% error.

Figure 3(b) provides a comparison of the number of function evaluations needed to find the solution around the optimum in the successful cases. It was found that the single agent needed at least four to five more function evaluations than the three agent system. In addition, it should be remembered that, potentially, the 3 agent algorithm could be run in parallel on three processors, which would further divide its computational cost by about 3.

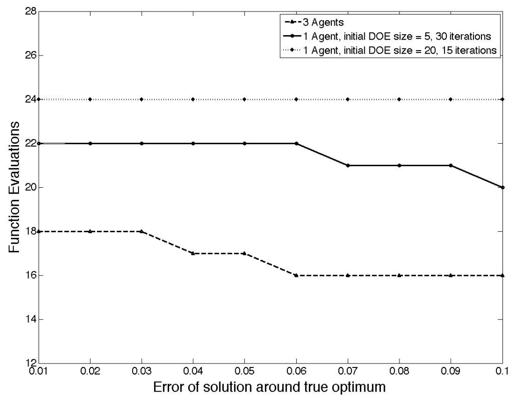
For the unsuccessful cases, we measured the difference between the feasible solution with the minimum objective function value from the true value at the optimum. Figure 4 displays this difference for the unsuccessful cases. It was found that the minimum average difference in f from the true optimum was approximately 0.4.

Figure 5 displays the mean value of the partial $PRESS_{RMS}$ divided by the mean data value of the subregion over the iterations for the surrogates of the objective function and the system-level reliability index. It was observed that the error in the surrogates decreased through the iterations after an initial increase in error in the first few iterations. The approximation of f was more accurate than of β_{sys}.

Figure 6 displays the subregions and approximations made of β_{sys} and f over 10 iterations for the three agent system for an initial DOE of 5 points. It was observed that the optimum was reached after 4 iterations. The role of agent 3 (represented in green) was primarily to explore. Agent 1 (in blue) explored until the 6th



(a) Number of successes in finding the true optimum



(b) Number of function evaluations

Figure 3. For 100 repetitions, (a) number of successes in finding true optimum, and (b) mean number of function evaluations needed to achieve a solution around the true optimum at some level of error

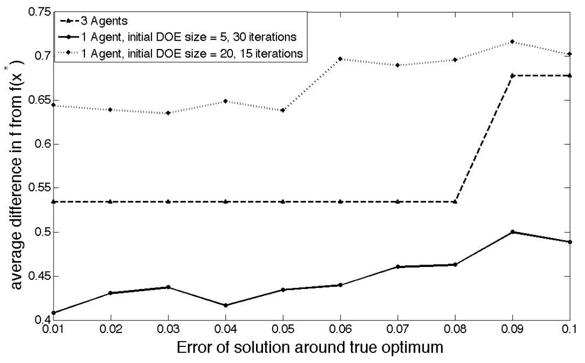


Figure 4. Examination of the unsuccessful cases in Fig.3(a) in which a solution at some error around the true optimum was not found: average difference of the feasible solution with minimum objective function value from $f(x^*)$

iteration, and after found local optima. By the end of the 10 iterations, the three agent system was able to provide good approximations of the true functions over the subregions.

The number of times each surrogate in Table 1 was used over the cumulative number of iterations for 100 repetitions is shown in Table 2. It was observed that moving least squares was used the most frequently for the surrogate of β_{sys} by the three agent system. However, for both cases of the single agent system, the most often used surrogate of β_{sys} was the quadratic response surface. Kriging (IDs 4 and 5 in Table 1) was the most favored for surrogate of the objective function in all cases.

3. Investigation of the Effect of Multiple Surrogates with Design Space Partitioning

The three agent system showed an inclination towards the surrogates of moving least squares for β_{sys} and kriging for f. To investigate the effectiveness of using multiple surrogates with design space partitioning, we fixed these surrogates for all agents, such that the agents no longer were required to choose a surrogate based on minimum $PRESS_{RMS}$. The same 100 initial DOEs were used with the three agent system and the number of successes in locating feasible solutions and solutions around the true optimum was measured.

In both configurations, the agents were able to locate feasible solutions in all repetitions. When the surrogates are fixed, the agents were only slightly more successful at locating the true optimum, doing so with usually fewer function evaluations as shown in Figs. 7(a) and 7(b). However, the sum of the partial

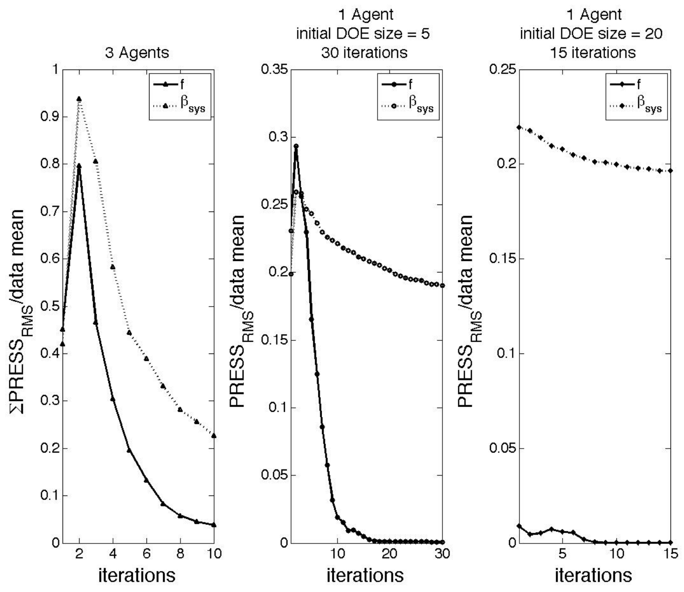


Figure 5. Mean $PRESS_{RMS}$ divided by data mean over 100 repetitions for 3 agents with an initial DOE of 5 points for 10 iterations (left), 1 agent an initial DOE of 5 points for 30 iterations (middle), and 1 agent with an initial DOE of 20 for 15 iterations (right). Note that the $PRESS_{RMS}$ for 3 agents is the sum of the partial $PRESS_{RMS}$ of the 3 subregions.

$PRESS_{RMS}$ errors was larger when the surrogates are fixed, as shown in Fig. 7(c). Therefore, it may not be possible to choose surrogates a priori based on $PRESS_{RMS}$, and having the choice of multiple surrogates may allow the agents to choose the surrogates that allow it to locate the optimum. In this case, in which the three agent system was highly successful, it was difficult to judge the effectiveness of using multiple surrogates in combination with the partitioning of the design space. We continued this investigation in the example problem in the following section.

B. Constrained Optimization with Disconnected Feasible Regions

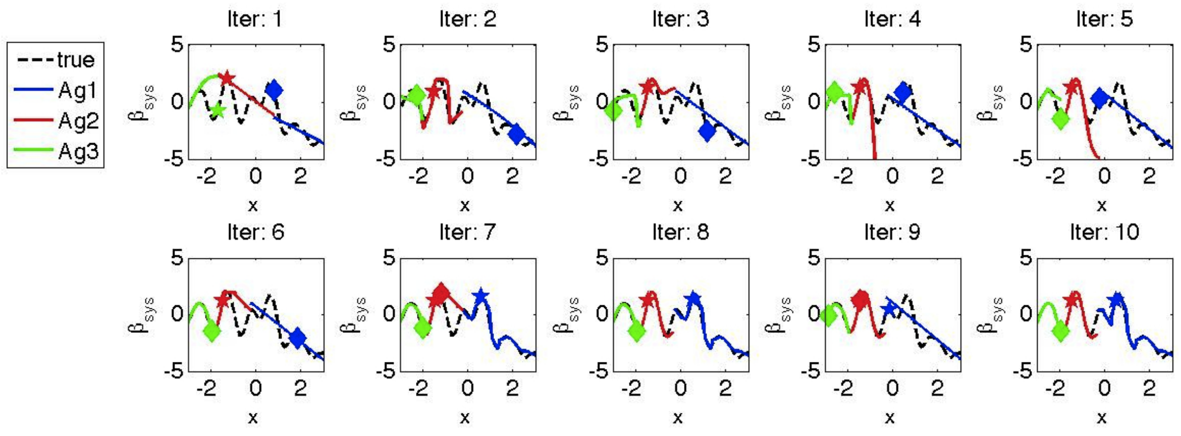
1. Problem Description

The second example is a two dimensional problem with a single constraint taken from Sasena et al.[12] The objective is quadratic and the single constraint is the Branin test function,[22] and is referred to as *newBranin*. This problem is shown in Eq.(5).

$$\begin{aligned}
\underset{x}{\text{minimize}} \quad & f(x) = -(x_1 - 10)^2 - (x_2 - 15)^2 \\
\text{subject to} \quad & g(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right) + \cdots \\
& 10\left(1 - \frac{1}{8\pi}\right) cos(x_1) + 10 - 2 \leq 0 \\
& -5 \leq x_1 \leq 10 \\
& 0 \leq x_2 \leq 15
\end{aligned} \tag{5}$$

The contour plot in Fig. 8 shows three disconnected feasible regions. These feasible regions cover approximately 3% of the design space. The global optimum is located at $x_1 = 3.2143$ and $x_2 = 0.9633$. There are also local optima located at (9.2153,1.1240) and (-3.6685,13.0299).

In this example, only the constraint g was considered to be the expensive function that would be approximated with surrogates. The seven possible surrogates, which include response surfaces, kriging, and moving least squares, are described in Table 1. The initial design of experiments of size 12 was found using Latin Hypercube sampling. Four agents were used to solve the problem.


(a) Subregions showing β_{sys} surrogates

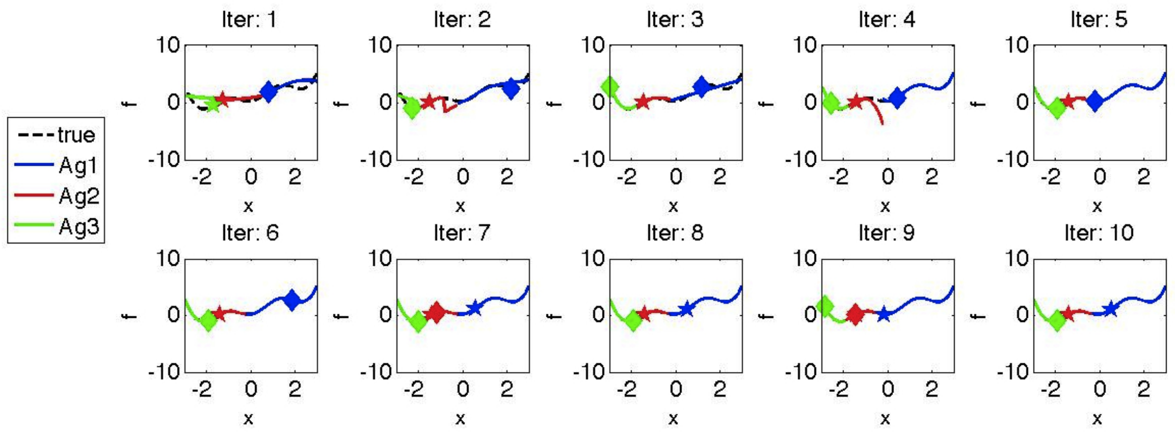

(b) Subregions showing f surrogates

Figure 6. Ten iterations for the three agent system showing the approximations given in each subregion for (a) β_{sys} and (b) the objective function f. The true functions are represented by the dashed black lines. The points added in each region are also shown (optima are represented by stars, exploration points by diamonds)

2. Results

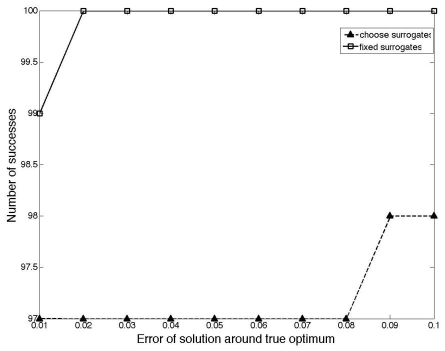
The processes described in Sec. II and III ran for $t^{max} = 20$ iterations with four agents, which is approximately 80 expensive function evaluations. This process was repeated for 100 different initial design of experiments. The results were compared to optimization using a single agent that optimized in the entire design space for 100 iterations, which is approximately 100 expensive function evaluations, and for 100 repetitions. The use of a single agent is equivalent to using a single surrogate to optimize locally but with exploration when the surrogate repeats points or provides infeasible points. This allowed a comparison of two processes with an almost equal number of function evaluations.

The four agent system was able to locate a feasible solution in all 100 repetitions, while the single agent system was successful in only 83 cases. Figure 9(a) compares the number of successes in 100 repetitions that the four agent system and single agent were able to locate a solution at some level of error around the true global optimum. For the smallest error of 1% around the optimum, the four agent system was successful 81 times. This 1% error is an error of approximately 0.03 in x_1 and 0.01 in x_2. When this error was increased to 10%, which is an error of approximately 0.3 in x_1 and 0.1 in x_2, the agents were successful in locating a solution 93 times. The system of four agents was much better at locating a solution compared to the single agent, which was only successful 53 times at a 10% error around the optimum.

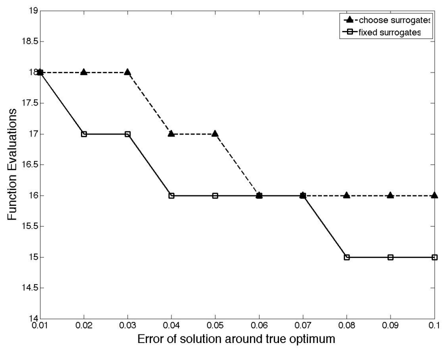
Figure 9(b) compares the number of function evaluations needed to find the solution around the global optimum in the successful cases. It was found that the single agent needed approximately ten fewer function evaluations than the four agent system. However, as shown in Fig. 9(a), the single agent was not as successful

Table 2. Number of uses of each surrogates considering each iteration and 100 repetitions. The surrogate ID corresponds to the surrogate described in Table 1.

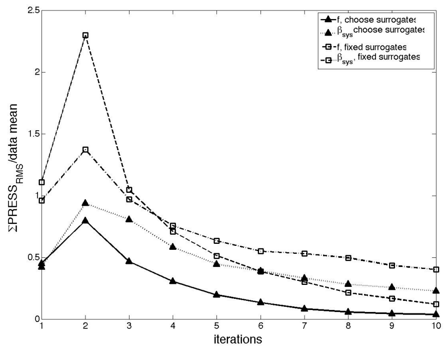
	3 Agents (3,000 cumulative iterations)		1 Agent, Initial DOE of 5 (3,000 cumulative iterations)		1 Agent, Initial DOE of 20 (1,500 cumulative iterations)	
Surrogate ID	β_{sys}	f	β_{sys}	f	β_{sys}	f
1	210	67	145	30	0	0
2	157	153	2446	82	1463	0
3	264	298	133	135	0	0
4	164	236	38	473	15	94
5	512	739	72	2111	4	1404
6	334	882	45	75	0	2
7	1359	625	121	94	18	0



(a) Number of successes in finding the true optimum



(b) Number of function evaluations



(c) $PRESS_{RMS}$

Figure 7. Comparing the three agent system when surrogates are chosen versus when surrogates are fixed for 100 repetitions: (a) number of successes in finding true optimum, (b) mean number of function evaluations needed to achieve a solution around the true optimum at some level of error, and (c) sum of the partial $PRESS_{RMS}$ over the subregions for f and β_{sys} surrogates

as the four agent system at finding the solution.

Due to the partitioning and search scheme in this method, agents should be able to locate the local optima. Figure 9(c) compares the number of successes in the 100 repetitions in finding all three local optima with some error about each optima. The four agent system is much more successful at locating all three

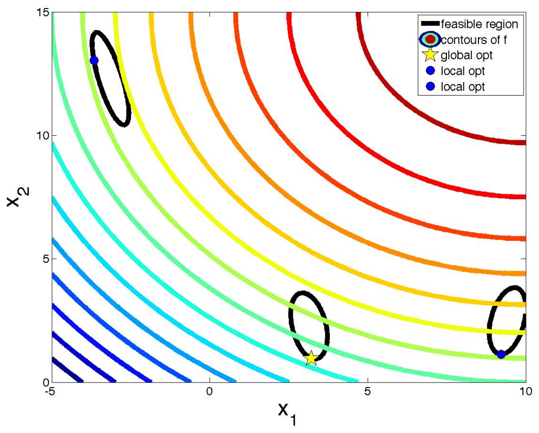
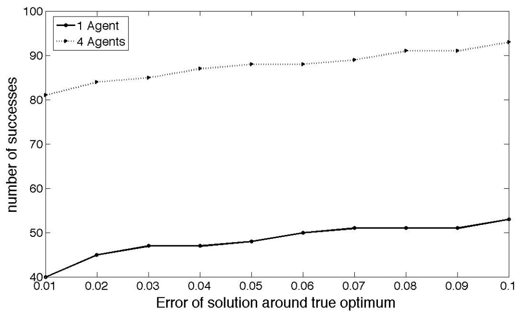
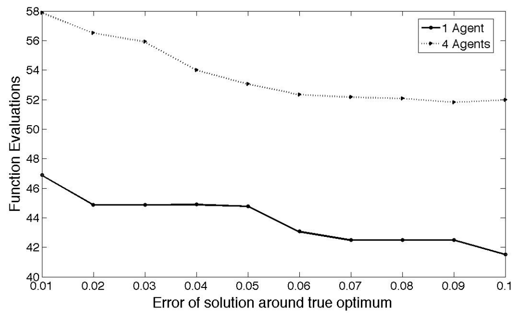


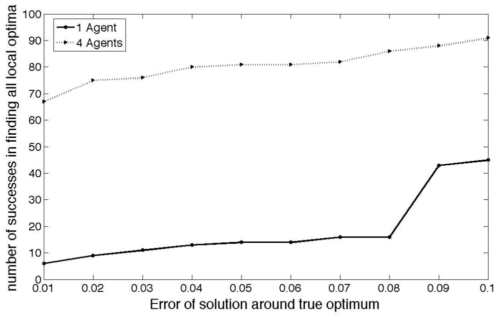
Figure 8. Contour plot of f for the $newBranin$ problem. The feasible region boundaries are shown as black lines and the global optimum is shown with a yellow star at $x_1 = 3.2143$ and $x_2 = 0.9633$. There are local optima located at (9.2153,1.1240) and (-3.6685,13.0299), represented by the blue circles.



(a) Number of successes in finding the true global optimum



(b) Number of function evaluations



(c) Number of successes in finding the local optima

Figure 9. For 100 repetitions, (a) number of successes in finding true global optimum, (b) mean number of function evaluations needed to achieve a solution around the true global optimum at some level of error, and (c) and number of successes in finding all three local optima

optima compared to the single agent. At 1% error around the optima, the four agent system located all three optima in 67 cases, whereas the single agent was successful in only 6 cases.

For the cases in which a solution around the true optimum was not found, we measured the difference between the feasible solution with the minimum value of f from the $f(x^*)$. This difference is shown in Fig. 10. There was around an average difference of around 35 to 55 between the best feasible solution and the true optimum of $f(x^*) = -243.1$.

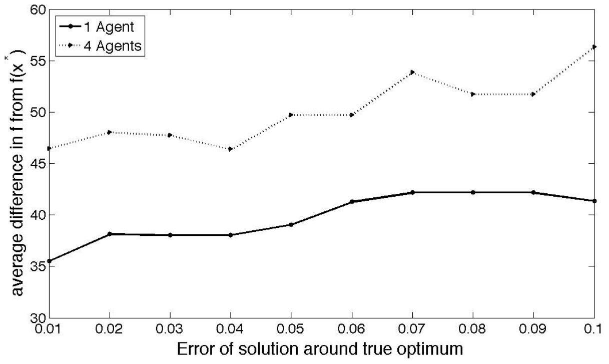


Figure 10. Examination of the unsuccessful cases in Fig.9(a) in which a solution at some error around the true optimum was not found: average difference of the feasible solution with minimum objective function value from $f(x^*)$

Figure 11 displays the mean value of the partial $PRESS_{RMS}$ divided by the mean data value of the subregion over the iterations. It was observed that the error in the surrogates chosen by the agents decreased through the iterations.

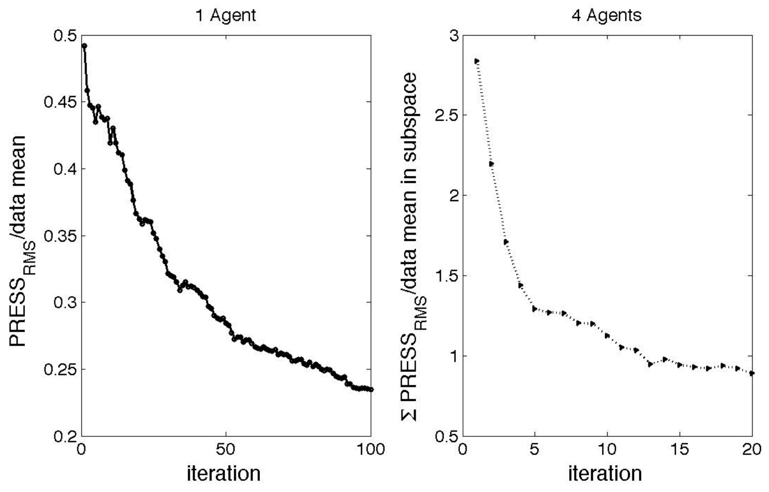
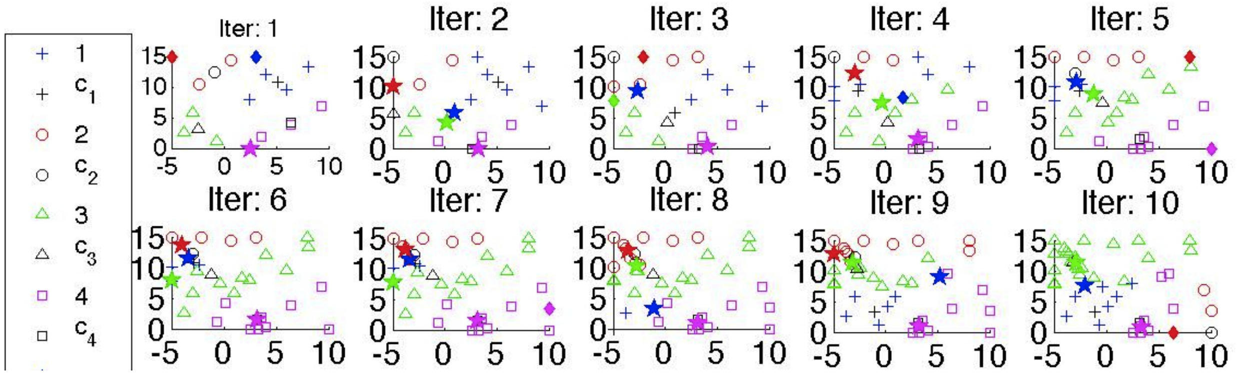


Figure 11. $PRESS_{RMS}$ divided by data mean over 100 iterations for 1 agent (left) and for 20 iterations for 4 agents (right). Note that the $PRESS_{RMS}$ for 4 agents is the sum of the partial $PRESS_{RMS}$ of the 4 subregions.

Figure 12(a) illustrates the the partitioning scheme described in this abstract on a random initial design of experiments. Figure 12(b) shows the agents' subregions. In the first iteration, the design space was partitioned using k-means clustering. In the first iteration, agents 1 and 2 (blue and red, respectively) were unable to locate an optimum, so the optimum of the exploration problem to maximize the minimum distance from existing points in the subspace resulted in the points represented by diamonds. Agents 3 and 4 (green and magenta, respectively) found the same optima, which is represented by the stars.

At the second iteration, the last optimum from the last iteration was an improvement for both agents 3 and 4, so the centroid of each agent should have been moved to that point. However, since these two points were too close together, agent 3 was deleted and moved to a point that maximized the minimum distance from the already existing centroids. In the same iteration, agent 1 saw no improvement so the centroid remained the same, whereas agent 2 saw some improvement so the centroid was moved to the last optimum.

Over the iterations, there was a clustering of points around the true global optimum by agent 4. At iteration 7, agent 4 provided an exploration point as it had found the same optimum in two consecutive iterations. The other agents were attracted to the other feasible region at $x_1 = -3.7$ and $x_2 = 13.0$. For


(a) Allocation of points and location of optima

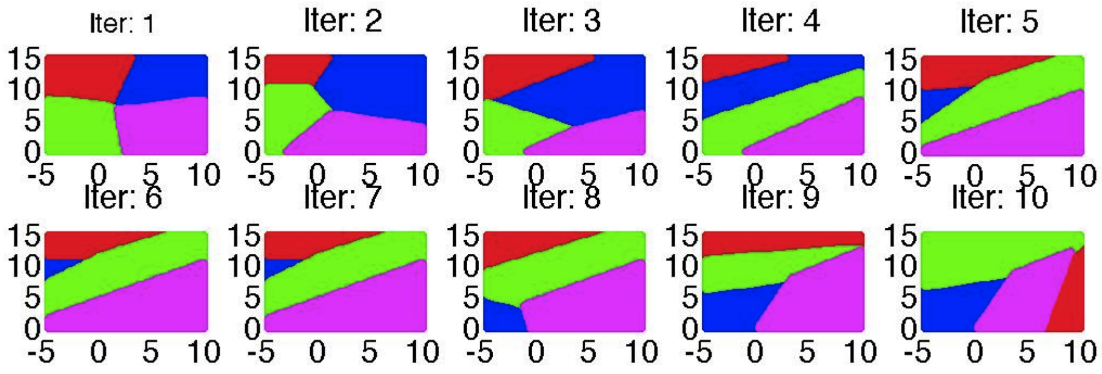

(b) Subregions

Figure 12. Ten iterations showing (a) allocation of design points, centroids, and added points (optima are represented by stars, exploration points by diamonds) and (b) the agents' subregions

some time, the other feasible region around $x_1 = 9.2$ and $x_2 = 1.1$ remained unexplored until agent 2 moved its subregion at iteration 10.

Table 3 the number of times each surrogate is used over the cumulative number of iterations over 100 repetitions. The surrogate that was used the most frequently was the cubic response surface, followed by the different kriging surrogates.

3. Investigation of the Effect of Multiple Surrogates with Design Space Partitioning

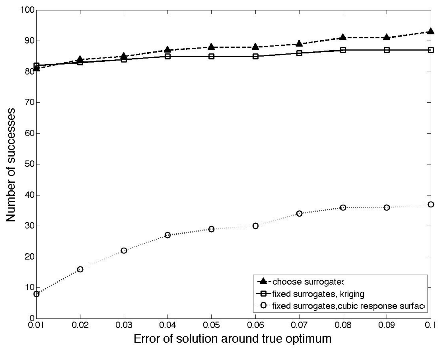
As in the previous example, we sought to measure the effect of using multiple surrogates with design space partitioning. This was done by fixing the surrogate that was used in the subregions. As the most frequently used surrogate was the cubic response surface (ID 3), we fixed this as a surrogate choice. As another comparison, we also fixed the surrogate at ID 4, a kriging surrogate.

The four agent system using kriging was able to locate a feasible solution in all of 100 repetitions, whereas the system using the cubic response surface was able to locate a feasible solution in 99 repetitions. The number of repetitions in which the four agent system was able to locate a solution within some error from the true optimum is given in Fig. 13(a). The system with the cubic response surfaces was clearly the least successful, with only 37 successes at 10% error around the optimum. The system that was allowed to choose from the set of surrogates slightly outperformed the system using kriging.

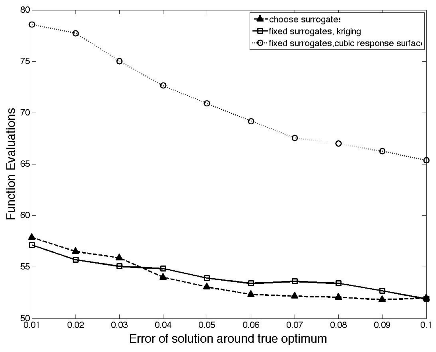
The number of function evaluations needed to reach the solution around the true optimum is displayed in Fig. 13(b), along with the sum of the partial $PRESS_{RMS}$ error over the subregions is shown in Fig. 13(c). The differences between the system with the choice of surrogate and the system with kriging were small. The system with the cubic response surfaces required approximately 30 more iterations, and had

Table 3. Number of uses of each surrogates considering each iteration and 100 repetitions

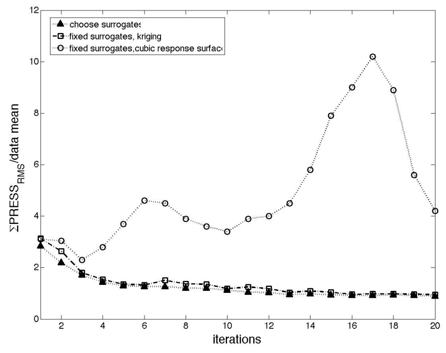
ID	4 Agents (8,000 cumulative iterations)	1 Agent (10,000 cumulative iterations)
1	241	0
2	789	30
3	2486	7881
4	1342	1085
5	1037	567
6	1117	415
7	988	22



(a) Number of successes in finding the true optimum



(b) Number of function evaluations



(c) $PRESS_{RMS}$

Figure 13. Comparing the four agent system when surrogates are chosen versus when surrogates are fixed for 100 repetitions: (a) number of successes in finding true optimum, (b) mean number of function evaluations needed to achieve a solution around the true optimum at some level of error, and (c) sum of the partial $PRESS_{RMS}$ over the subregions

much larger errors that did not follow the trend of decreasing through the iterations.

These results support the idea that the agents should choose a surrogate from a set of surrogates rather than fix a single surrogate a priori. As seen in this example, the choice of which surrogate is fixed can have a large effect on the success of the agents. Conversely, we observed success in both examples when kriging surrogates were the fixed surrogates. It is possible that kriging leads to better results since it interpolates the data points, compared to polynomial response surfaces which follow the trend of the data. It is not clear whether constraining all agents to have the same surrogate will also have a detrimental effect.

V. Conclusions and Future Work

The work shown in this article displayed the efficiency and reliability of a multi-agent optimization method. Agents were defined as surrogate-based optimization tasks whose actions were coordinated by partitioning the design space around local optima. Because each agent searches a smaller space, it solves an easier optimization problem. The proposed partitioning algorithm aims at balancing agents' searches in the whole design space and stabilizing agents around the best found local optima. It was shown that this method has the ability to find the global optimum.

At this point, the agents are working together through a global database and partitioning at each cycle. In the future, we will investigate methods of decentralization and asynchronization, such that the partitioning and updating of the global database does not occur at every iteration allowing each agent to work in its own time. Other future work includes studying the effectiveness of this method on higher dimensional problems and finding how to allocate resources with an optimal number of agents.

Acknowledgments

This work has benefited from funding from Agence Nationale de la Recherche (French National Research Agency) with ANR-09-COSI-005 reference.

References

[1]Villanueva, D., Le Riche, R., Picard, P., and Haftka, R. T., "A Multi-Agent System Approach To Reliability Based Design Optimization Including Future Tests," *12e Congrès de la Société Francaise de Recherche Opérationnelle et d'Aide à la Decision (ROADEF'11)*, Saint-Etienne, France, 2011.

[2]Jin, R., Du, X., and Chen, W., "The use of metamodeling techniques for optimization under uncertainty," *Structural and Multidisciplinary Optimization*, Vol. 25, No. 2, 2003, pp. 99–116.

[3]Queipo, N. V., Haftka, R. T., Shyy, W., and Goel, T., "Surrogate-based analysis and optimization," *Progress in Aerospace Sciences*, Vol. 41, 2005, pp. 1–28.

[4]Sacks, J., Welch, W. J., J., M. T., and Wynn, H. P., "Design and analysis of computer experiments," *Statistical Science*, Vol. 4, No. 4, 1989, pp. 409–435.

[5]Simpson, T. W., Peplinski, J. D., Koch, P. N., and Allen, J. K., "Metamodels for computer based engineering design: survey and recommendations," *Engineering with Computers*, Vol. 17, No. 2, 2001, pp. 129–150.

[6]Voutchkov, I. and Keane, A. J., "Multiobjective optimization using surrogates," *7th International Conference on Adaptive Computing in Design and Manufacture*, Bristol, UK, 2006, pp. 167–175.

[7]Samad, A., Kim, K., Goel, T., Haftka, R. T., and Shyy, W., "Multiple surrogate modeling for axial compressor blade shape optimization," *Journal of Propulsion and Power*, Vol. 25, No. 2, 2008, pp. 302–310.

[8]Viana, F. A. C. and Haftka, R. T., "Using multiple surrogates for metamodeling," *7th ASMO-UK/ISSMO International Conference on Engineering Design Optimization*, Bath, UK, 2008.

[9]Glaz, B., Goel, T., Liu, L., Friedmann, P., and Haftka, R. T., "Multiple-surrogate approach to helicopter rotor blade vibration reduction," *AIAA Journal*, Vol. 47, No. 1, 2009, pp. 271–282.

[10]Zhao, D. and Xue, D., "A multi-surrogate approximation method for metamodeling," *Engineering with Computers*, Vol. 27, 2005, pp. 139–153.

[11]Wang, G. G. and Simpson, T. W., "Fuzzy Clustering Based Hierarchical Metamodeling for Design Space Reduction and Optimization," *Engineering Optimization*, Vol. 36, No. 3, 2004, pp. 313–335.

[12]Sasena, M., Papalambros, P., and Goovaerts, P., "Global Optimization of Problems with Disconnected Feasible Regions Via Surrogate Modeling," 2002.

[13]Jones, D. R., Schonlau, M., and Welch, W. J., "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, Vol. 13, No. 4, pp. 455–492.

[14]Venkatraman, S. and Yen, G., "A generic framework for constrained optimization using genetic algorithms," *IEEE Trans Evol Comput*, Vol. 9, 2005, pp. 424–435.

[15]Fletcher, R. and Leyffer, S., "Nonlinear programming without a penalty function," *Math Program*, Vol. 91, No. 2, 2002, pp. 239–269.

[16]Liu, C., "New multiobjective PSO algorithm for nonlinear constrained programming problems," *Advances in cognitive neurodynamics ICCN 2007*, Springer, New York, 2007, pp. 955–962.

[17]Zhou, Y., Li, Y., He, J., and Kang, L., "Multi-objective and MGG evolutionary algorithm for constrained optimization," *The 2003 Congress on Evolutionary Computation*, Canberra, 2003, pp. 1–5.

[18]Venter, G. and Haftka, R., "Constrained particle swarm optimization using a bi-objective formulation," *Structural and Multidisciplinary Optimization*, Vol. 40, 2005, pp. 64–76.

[19]Aurenhammer, F., "Voronoi diagrams: a survey of a fundamental geometric data structure," *ACM Computing Surveys (CSUR)*, Vol. 23, No. 3, 1991, pp. 345–405.

[20]Hartigan, J. and Wong, M., "Algorithm AS 136: A K-Means Clustering Algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, Vol. 28, No. 1, 1979, pp. 100–108.

[21]MATLAB, *version 7.9.0.529 (R2009b)*, chap. fmincon, The MathWorks Inc., Natick, Massachusetts, 2009.

[22]Dixon, L. and Szego, G., *Towards Global Optimisation 2*, chap. The Global Optimisation Problem: An Introduction, North-Holland Publishing Company, New York, 1978.