Development of a Hybrid Metamodel based Simulation Optimization Algorithm

By Farhad Ghassemi Tari & Zohreh Omranpour

Sharif University of Technology, Iran

Abstract- In this paper, a metamodel based hybrid algorithm was developed for optimization of digital computer simulation models. The simulation models are considered to be computationally expensive. It is also considered to have a single stochastic and unconstrained response function. The hybrid algorithm is developed by modification and integration of several concepts and routines. We employed the nested portioning and the particle swarm optimization algorithms to develop an efficient search mechanism for the hybrid algorithm. Then we integrated the modified Kriging metamodel to the search mechanism for facilitating the function fitting processes of the simulation’s output. The efficiency of the developed hybrid algorithm was then evaluated through computational experiments. Ten complex test problems were selected from the literatures and the efficiency of the developed hybrid algorithm was evaluated by comparing its performances against three known algorithm which are cited in the literature. The result of these computational experiments revealed that the developed hybrid algorithm can provide very robust solutions with a very low computational effort.

Keywords: simulation, optimization, nested partitioning, stochastic kriging, particle swarm optimization.

GJRE-G Classification : FOR Code: 290502p

Strictly as per the compliance and regulations of:
Development of a Hybrid Metamodel based Simulation Optimization Algorithm

Farhad Ghassemi Tari \(^a\) & Zohreh Omranpour \(^a\)

Abstract- In this paper, a metamodel based hybrid algorithm was developed for optimization of digital computer simulation models. The simulation models are considered to be computationally expensive. It is also considered to have a single stochastic and unconstrained response function. The hybrid algorithm is developed by modification and integration of several concepts and routines. We employed the nested portioning and the particle swarm optimization algorithms to develop an efficient search mechanism for the hybrid algorithm. Then we integrated the modified Kriging metamodel to the search mechanism for facilitating the function fitting processes of the simulation’s output. The efficiency of the developed hybrid algorithm was then evaluated through computational experiments. Ten complex test problems were selected from the literatures and the efficiency of the developed hybrid algorithm was evaluated by comparing its performances against three known algorithm which are cited in the literature. The result of these computational experiments revealed that the developed hybrid algorithm can provide very robust solutions with a very low computational effort.

Keywords: simulation, optimization, nested partitioning, stochastic kriging, particle swarm optimization.

I. INTRODUCTION

Digital computer simulation models have been very successful approach for analyzing the complex systems. In simulation models the analytical expression for input/output relationship is generally unavailable and hence conducting its output performance analysis is a cumbersome task. Based on Barton and Meckesheimer [2] simulation optimization is defined as a repeated analysis of simulation models with different values of design parameters, in an attempt to identify best simulated system performance. Since 1950 when simulation optimization has been introduced as a new area for research, many practical problems are modeled as simulation optimization problems and successful results have been achieved. Some recent achievements are the works of Liu and Maghsoodloo, [17] and Kleijnen et al.[14]. Also a number of software packages have been developed with the ability of simulation optimization and have been added to some well-known simulation software’s. However, the related literatures are still waiting for developments of more new robust methods with the high results efficiency. For recent surveys in this field, we acknowledge the research studies conducted by Fuand Glover [7], Tekin and Sabuncuoglu [28], Henderson and Nelson [8] and Kleijnen [13].

Since simulation models have stochastic response they potentially require extensive runtime. When an optimization routine is incorporated to the simulation models, this problem will intensify dramatically. Due to the stochastic nature of the simulation models, the output must be estimated by averaging of the outputs values over reasonable number of replications [10]. In addition the simulation models are usually considered as the black-box models in which, the output function is not usually expressed explicitly. Therefore determining the best combination of controllable variables which provide the best output value can be done either by fitting an explicit function to the output values or by one of the algorithmic input-output optimization routines.

One of the approaches is the use of metamodel-based methods. Metamodels provide deterministic objectives, which surrogate simulation models, and generally need fewer computational efforts. One of the powerful metamodel, addressed in the literatures, is Kriging method. Kriging is an interpolation method which initially developed for using in spatial statistics. This method which is initially proposed by Krige [15] was applied by Cressie [5] in geology. Later it was applied in areas like economic and modeling black-box computer experiments [9].

The efficiency of this method in deterministic simulation models has proven [11,20,26]. Based on the good results obtained in deterministic simulation, Beer and Klijnen [3] applied this method to the random simulation models. With considering successful application of Kriging methodology in design and analysis of deterministic computer experiments Ankenman et al [1]. Introduced stochastic Kriging (SK) metamodel inspired by Kriging methodology. Actually they extend Kriging methodology which applied successfully in deterministic simulation as a global metamodel, to the stochastic simulation. Despite of high accuracy and successful results, SK has not been use as metamodel in numerous papers and has not been use to any of the commercial simulation software package.

A powerful population based metaheuristics which can be employed for solving large-scale optimization problems efficiently is nested partition (NP) method. This method introduced by Shi and Ölfsson...
In this paper we developed a metamodel-based simulation optimization algorithm. The developed algorithm is hybrid by the fact that it is constructed by modification and combination of several optimization routines. We used stochastic Kriging (SK) metamodel for fitting a functional relation to the input output of our simulation models. SK actually is a global metamodel, for fitting a functional relation to the input output of our simulation models has been very successful. Examples are inventory management simulation, inventory production simulation models in logistics and operation management and queueing simulation models.

This problem is formulated as follow:

$$\min_x E(w(x))$$

$$l_j \leq x_j \leq u_j \quad j = 1, 2, ..., d$$

Where \( x = [x_1, x_2, ..., x_d]^T \) denotes a \( d \)-dimensional input vector and \( w \) indicates the output value. The expected value in the objective function is estimated by simulation and we consider only box constraints where \( l_j \) and \( u_j \) are the lower and upper bounds of the input variables \( x_j \) respectively.

III. Development of the Hybrid Algorithm

The developed hybrid algorithm is a metamodel-based optimization process. We used the stochastic Kriging (SK) metamodel and employed a sequential experiment design for validating its results. In optimization part we used the hybrid metaheuristic algorithm of nested partition (NP) method, which benefits of quick convergence for large scale problems with high computational efforts and we also used the advantages of Particle swarm optimization (PSO) algorithm for its simplicity and good local search ability.

During the process of developing the hybrid algorithm, we first developed an algorithm with the integration of the NP and the PSO algorithm, and we called it PSPO algorithm. The efficiency of the PSPO algorithm was then tested through a computation experiment. As the result, we found that although the solution obtained by this algorithm was either optimal or closed to the optimal, but the computational efforts to obtain the solution were extensive. Due to this problem we directed the path of our research to the metamodel based algorithm and we developed the hybrid algorithm.

Fig. 1 represents a flow diagram of the developed hybrid algorithm. According to Fig. 1, the first part of the hybrid algorithm includes initial sampling (step 1) and simulating this sample points with a number of specified iteration (step 2). The second stage
is an iterative process for fitting metamodel using the input output data (step3) and then validating metamodel by the method developed by Liu, Nelson, and Staum [18]. Their method is actually based on cross-validation method and developed for stochastic Kriging metamodels (step4). Finally the last stage, algorithm starts with primary partitioning of feasible space (step 5) and then sampling and improving these sample points in each partition (step6). By using the validated metamodel and improved samples we computed the fitness value of each partition and then determine the best partition (the most promising partition) and consequently the best point (step7). Actually this point was used for re-partitioning the solution space and adds the best point (step7) and fitness value of each partition and then determine the most promising partition by $\Theta$. Now, the following sections provide more details for each step.

Step 1 : Performing the initial experiment design

The first step of the hybrid algorithm includes generating some initial simulation observations, based on a statistical sampling. Selecting an efficient experiment design is an important step in the process of developing a metamodel and can affect the hybrid algorithm performances.

Since we intend to use stochastic Kriging metamodel and based on the assumption that there is no former information about output values, we employed the max-min Latin Hyper Cube sampling [13] while checking the box constraints. Considering a $d$-dimensional space, we used the sample size suggested by Kleijnen [14]. Therefore the sample size was designated as $5 + 2d$ of the initial sample points. In step 5 we will explain more detail regarding the sequential experiment design.

Step 2 : Simulating the initial design points

Stochastic simulation models have stochastic output values, so we should obtain an average of simulation outputs over the number of replications. If we denote $n_{i}$ as the number of simulation runs in a design point $x_{i}$, the average will be obtained by:

$$\bar{w}(x_{i}) = \frac{\sum_{j=1}^{n} w_{j}(x_{i})}{n_{i}}$$ (3)

Where $w_{j}(x_{i})$ represent simulation output in the $j^{th}$ iteration. For determining the number of simulation run (replication) in each design point, we used the method of Liu et al. [18] through which the output variance is reduced. At first we performed $n_{0}$ replications for every design point $x_{i}$. Then we calculated mean $\bar{w}(x_{i})$ and the variance $S^{2}(x_{i})$ of these replications. We targeted a relative precision for simulation output $\gamma$ and then determined the total replication size $n_{i}$ in each design point. The relative precision for $n_{i}$ replication in design point $x_{i}$ was calculated by

$$\frac{l(x_{i}, n_{i}, \alpha)}{\bar{w}(x_{i})}$$ which we required it to be less than $\gamma$. Where $l(x_{i}, n_{i}; \alpha)$ is a half-width of the $(1-\alpha)$ confidence interval for output $\bar{w}(x_{i})$. We then used the following condition, proposed by Law [16], for obtaining $n_{i}$:

$$\frac{l(x_{i}, n_{i}; \alpha)}{\bar{w}(x_{i})} \leq \frac{\gamma}{1 + \gamma} \quad (4)$$

And therefore $n_{i}$ is obtained as:

$$n_{i} = \max \left\{ n_{0}, \left[ \frac{(1 + \gamma)\bar{w}(x_{i})}{\gamma S(x_{i})} \right] \right\}$$ (5)

As the result, $n_{i} - n_{0}$ more replication runs in the design point $x_{i}$ should be executed. After determining design points and number of replication in each design point, simulation of all the design points can be executed.

Step 3 : Fitting stochastic Kriging metamodel

By using initial design points and simulation model output (I/O data) we fit stochastic Kriging metamodel, and used it for estimation of the simulation output. Considering $m$ initial design points, the stochastic Kriging predictor which optimizes mean square error (MSE) has the form of:
\[ \hat{w}(x_c) = \hat{\beta}_0 + \hat{\tau}^2 R_M(x_c, \cdot, \hat{\theta})^T \left[ \hat{\tau}^2 R_M(\hat{\theta}) + \hat{\Sigma}_e \right]^{-1} (\mathbf{w} - \hat{\beta}_0 1_m) \]  

(6)

**Figure 1:** The flow diagram of the hybrid algorithm

Where \( w_s \) is a point to be predicted, \( R_M \) is the correlation between two points which depends on the distance between points and a function of \( \hat{\theta}, \hat{\Sigma}_e \) is an \( m \times m \) matrix and is calculated by the following formula:

\[
\hat{\Sigma}_e = \text{Diag} \left\{ \frac{\hat{V}(x_1)}{n_1}, \frac{\hat{V}(x_2)}{n_2}, \ldots, \frac{\hat{V}(x_m)}{n_m} \right\}
\]

where

\[
\hat{V}(x_i) = \frac{1}{n_i-1} \sum_{j=1}^{n_i} (w_j(x_i) - \bar{w}(x_i))^2,
\]

\( \hat{\tau}^2, \hat{\theta}, \hat{\beta}_0 \) are estimated by solving the following likelihood equation:

\[
i(\beta_0, \tau^2, \theta) = -\ln [(2\pi)^{m/2}] - \frac{1}{2} \ln \left[ \hat{\tau}^2 R_M(\theta) + \Sigma_e \right] - \frac{1}{2} (\mathbf{w} - \beta_0 1_m)^T \left[ \hat{\tau}^2 R_M(\theta) + \Sigma_e \right]^{-1} (\mathbf{w} - \beta_0 1_m),
\]  

(7)
and \( \bar{w} = (\bar{w}(x_1), \bar{w}(x_2), \ldots, \bar{w}(x_m))^T \), where \( \bar{w}(x_i) \) calculated by equation (3) and \( I_m \) is a vector of 
ones with a compatible dimension.

For more details, we refer the reader to Ankenman, Nelson, and Staum [1].

**Step 4: Validating of the Metamodel**

For checking the validity of metamodel we used 
the leave-one-out cross-validation method. The main goal of this method is to decrease the difference 
between real simulation output in a specific design 
point, \( w(x_i) \), and the metamodel prediction for the same 
design point, after eliminating the point designated by 
\( \hat{w}^{(i-1)}(x_i) \). In other words, the key idea is to control the 
relative leave-one-out prediction error at each design 
point which is calculated as follow:

\[
\frac{\hat{w}^{(i-1)}(x_i) - w(x_i)}{|w(x_i)|} \tag{8}
\]

Liu and coworkers (2010) demonstrated achieving this goal need to control \( E_i \) value which is 
calculated as follow:

\[
E_i = \frac{l(x_i, n_i; \alpha)}{|\bar{w}(x_i)|} + \frac{|\hat{w}^{(i-1)}(x_i) - \bar{w}(x_i)|}{|\bar{w}(x_i)|} - l(x_i, n_i; \alpha) \tag{9}
\]

Where the half-width is a measure of uncertainty in \( \bar{w}(x_i) \).

Before cross-validation we should separate design points which are not on the edges, i.e. for design 
point \( x_i = (x_i, x_{i2}, \ldots, x_{id}) \) in the \( d \) dimensions, we select 
point which none of the factors contains the extreme value. This subset of designing points is showed by set \( II \) and \( E_i \) is calculated for this subset.

Based on equation (9) the lack of credibility of \( E_i \) originated from two sources:

- Relative difference between simulation output and true value (relative precision of simulation output) which need to add more replication.
- Relative difference between true value and leave-one-out prediction which need to add more design point.

In each iteration, if \( E_i > \beta \), even just for one design point, we should add some more design points or add more replication to existing design points. If the 
first term of \( E_i \) is less than \( \lambda \beta \), we select a new design point and we add it to the experimental design points. Otherwise, we add more simulation replications for the point with maximum value of \( E_i \). This selection is based on covering complex areas, when \( \lambda \) is a predefined parameter which aims to control the effect of Mont Carlo variability during cross-validation [17] for \( \lambda \in (0, 1) \). We selected \( \lambda = 1/4 \) in our experiments.

The procedure for the validation process can be summarized by the following steps:

1. Separate design points which are not on the edges and show this subset by \( II \).

2. Determine number of simulation replication \( n_i \) in each design points as in equation (5) and set \( N_i \leftarrow n_i \).

3. Calculate \( \bar{w}(x_i) \), and \( S^2(x_i) \) for all design points in set \( Ii \).

4. Calculate \( E_i \) as in equation (9) for all design point in set \( Ii \).

5. \( i^* \leftarrow \arg \max_{i \in Ii} E_i \).

6. If \( E_{i^*} > \beta \),
   6.1. If \( \frac{l(x_{i^*}, n_{i^*}; \alpha)}{|\bar{w}(x_{i^*})|} - l(x_{i^*}, n_{i^*}; \alpha) > \lambda \beta \)
      a) Run simulation model at \( x_{i^*} \), \( N_{i^*} \) more replication and set \( N_{i^*} \leftarrow 2N_{i^*} \).
      b) Calculate \( \bar{w}(x_{i^*}) \), and \( S^2(x_{i^*}) \) again.
      c) Go to step 4.
   6.2. Else
      a) Add a new design point in the middle of distance between \( x_{i^*} \) and the nearest design point.
      b) Calculate the number of simulation replication in new design point and then calculate \( \bar{w}(x_{k+1}) \), and \( S^2(x_{k+1}) \).
      c) \( k \leftarrow k + 1 \).
      d) Go to step 4.

The main advantage of this validation method is considering the possibility of adding simulation replications in some points which has high error.
Step 5: Initial partitioning

In optimization process first we should partition experimental area. Different methods have been developed for partitioning in the NP algorithm. In this article for initial partitioning we divided the range of each variable, which is defined by the box constraint, into two halves. Through this way each variable’s dimension is divided into two equal parts. Then we use the SK algorithm for determining the potential areas.

Step 6: Sampling and improving samples

After defining partition boundaries, a uniform random sampling procedure was used for each partition. In this article the PSO algorithm is used to improve the quality of initial samples and generate feasible solutions which improve the effectiveness and efficiency of the NP algorithm. Suppose we have \( N \) initial random samples in the \( \text{j}^{th} \) partition denoted by \( D_j^j = \{x_{1j}^j, x_{2j}^j, \ldots, x_{Nj}^j\} \). Using PSO algorithm, the hybrid algorithm generates a sequence of improving solutions until no further improvement is possible. The final solutions which are more likely to be near optimal solution is shown by \( D_F^j = \{x_{1F}^j, x_{2F}^j, \ldots, x_{NF}^j\} \).

\[
Y(\sigma_j) = \min_{i \in [1,2,\ldots,N]} Y(x_{Fi}^j) \quad j = 1,2,\ldots,5
\]  

(10)

And the partition with the best overall fitness coefficient is chosen as the next most promising partition, with the index as follow:

\[
\hat{J}_k \leftarrow \arg Y(\sigma_j) \quad j = 1,2,\ldots,5
\]  

(11)

If this index corresponds to a sub region of \( \sigma(k), \hat{J}_k \leq 4 \), we let this to be the next most promising partition, ie:

\[
\sigma(k + 1) = \sigma_{\hat{J}_k}(k)
\]  

(12)

But if it belongs to complementary region we backtrack to previous iteration, ie:

\[
\sigma(k + 1) = \sigma(k - 1)
\]  

(13)

All partitions features in different iterations are saved. Also the backtrack-flag is set on. This flag is used for counting the number of back tracking simulation runs.

\[
\hat{J}_i = \arg \min_{i \in [1,2,\ldots,N]} x_{Fi}^j \quad j = 1,2,\ldots,5
\]  

(14)

So the best overall point will be denoted by \( x_{k_{\text{opt}}}^\text{opt} = x_{F_{\hat{J}_{\text{opt}}}}^\text{opt} \) which will be used in next partitioning (step 8). We also record the best point which is obtained so far \( x_{k_{\text{opt}}}^\text{opt} \) and we record the point which has the better performance measure.

Step 7: Determining winner partition and near optimal solution

Final population is used for estimating promising index and finally determining the winner partition (next most promising region). To achieve this goal we use the validated metamodel to estimate the output for all the final samples which are shown by \( Y(x_F^j) \) for \( f^{th} \) sample in \( j^{th} \) partition.

The best answer for each partition shows fitness coefficient for that partition. For example in minimization problems it is calculated as follow:

By using high quality samples we increase the probability of selecting correct partition and making correct moves. Assuming simulation runs are computationally very time consuming, in the process of running the PSO, we used the metamodel for estimating outputs instead of using the simulation run outputs.

It should be noted that in the next iterations (except first iteration) we will have five partitions for sampling. More details provide in step8.

Step 8: Updating the metamodel and re-partitioning solution space

In this step the near-optimal point which is obtained from previous step, \( x_{k_{\text{opt}}}^\text{opt} \), is simulated and then it will be added to the experimental design. With this new experimental design we re-fit and re-validate the metamodel. This new metamodel which may be
more likely to have better prediction is used in the next iteration.

As we explained before, if the current near-optimal point belongs to the complementary region we back track and we use previous partitioning, but if this point belongs to the promising region, the algorithm partitions the promising region into four new partitions (four rectangles) somehow that the obtained best point becomes one of the corner of these rectangular while the other corners are kept as the corners of the promising region. It is clear that these regions may not have equal area. All the remaining parts of the region except these four partitions are aggregated together and will be formed as the complementary partition.

Having these five partitions, the algorithm checks the stopping rule and if the stopping rule is not satisfied returns back to step 6 for the new sampling.

**Step 9 : Checking stopping rule**

The hybrid algorithm is iteratively proceeds until no significant improvement is realized. In order to check this condition the algorithm compares the objective function value of the current best point, \( \bar{w}(x^{opt}_k) \), with the best point obtained among all the previous iterations, \( \bar{w}(x^{opt}) \). To conduct this it first calculate the student prediction error as follow:

\[
t_0 = \frac{\bar{w}(x^{opt}_k) - \bar{w}(x^{opt})}{\sqrt{\bar{w}(x^{opt}_k)} + \sqrt{\bar{w}(x^{opt})}}
\]  

(15)

The current best point is accepted as the best solution point until now, only if \( t_0 < t_{1-\alpha, df} \), otherwise, we conclude that there was not any improvement, where is the degree of freedom and is calculated as follow:

\[
df = \min(n_k, n_{opt})
\]  

(16)

In our experiments we set \( \alpha = 0.1 \). We also set a threshold for maximum number of unimproved replications equal to 30, i.e. \( n_{max} = 30 \). Whenever this threshold is reached, the iterations are terminated. The algorithm is stopped and the best solution point is acknowledged as the final solution point \( (x^{opt}) \).

**IV. Computational Experiments**

For evaluating the performances of the developed algorithm, it is common to employ some known response surface functions, which are presented in the optimization literatures. We selected 10 complex test problems as the benchmark. Due to deterministic nature of these problems, we add a noise to their objective function relations. The main advantage of using these test problems was their complexity of obtaining their optimal solutions.

In this section the developed hybrid algorithm is evaluated against three known algorithms cited in the literatures. Ten unconstrained hard test problems with the following mathematical structures were selected from the literatures.

1. \( P_1 \); which has just one global minimum, with the following mathematical form:

\[
P_1 = \frac{x_1}{(1 + x_1^2 + x_2^2)} - 2 \leq x_j \leq 2 \quad j = 1, 2
\]  

(17)

2. \( P_2 \); which has one local minimum and one global minimum with the following mathematical form:

\[
P_2 = 3(1 - x_1^2)\exp(-x_1^2 - (x_2 + 1)^2) - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right)\exp(-x_1^2 - x_2^2) - \left(\frac{1}{3}\right) - \exp(-x_1^2 - x_2^2)
\]  

\[-3 \leq x_j \leq 3 \quad \text{for} \ j = 1, 2
\]  

(18)

3. \( P_3 \); which has two local minimum and one global minimum, with the following mathematical form:

\[
P_3 = -10(1 - x_1^2)\exp(-x_1^2 - (x_2 + 1)^2) - 15\left(\frac{x_1}{5} - x_1^3 - x_2^5\right)\exp(-x_1^2 - x_2^2) - \left(\frac{1}{3}\right)
\]  

\[-\exp(-x_1^2 - x_2^2)
\]  

\[-3 \leq x_j \leq 3 \quad \text{for} \ j = 1, 2
\]  

(19)

4. \( P_4 \); which is called Six Hump Camel back (SHC) has six local minimum and two global minimum, with the following mathematical form:
\[ P_4 = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + \left(-4 + 4x_2^2\right)x_2^2 \quad \text{for } j = 1,2 \]

5. P5; which called Himmelblau function (Hmb) and has several local minimum and one global minimum with the following mathematical form:

\[ P_5 = \left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2 + 0.1\left((x_1 - 3)^2 + (x_2 - 2)^2\right) \]
\[ -6 \leq x_j \leq 6 \quad \text{for } j = 1,2 \]

6. P6; which has several local minimum and one global minimum, with the following mathematical form:

\[ P_6 = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_2) + 0.7 \quad -100 \leq x_j \leq 100 \quad \text{for } j = 1,2 \]

7. P7; which called Booth function and has several local minimum and one global minimum with the following mathematical form:

\[ P_7 = \left(x_1 + 2x_2 - 7\right)^2 + \left(2x_1 + x_2 - 5\right)^2 \quad -100 \leq x_j \leq 100 \quad \text{for } j = 1,2 \]

8. P8; which called Michalewics function and has several local minimum and one global minimum with the following mathematical form:

\[ P_8 = -\sin(x_1)\sin^{20}\left(x_1^2/\pi\right) - \sin(x_2)\sin^{20}\left(2x_2^2/\pi\right) \]
\[ -100 \leq x_j \leq 100 \quad \text{for } j = 1,2 \]

9. P9; which called Sphere function and has no local minimum except the global one with the following mathematical form:

\[ P_9 = x_1^2 + x_2^2 - 5.12 \leq x_j \leq 5.12 \quad \text{for } j = 1,2 \]

10. P10; which called Brownian function and has three global minimum with the following mathematical form:

\[ P_{10} = \left(x_2 - \left(\frac{5}{4\pi}\right)x_1^2 + \left(\frac{5}{\pi}\right)x_1 - 6\right)^2 + 10\left(1 - \left(\frac{1}{8\pi}\right)\right)\cos(x_1) + 10 \]
\[ -5.12 \leq x_j \leq 5.12 \quad \text{for } j = 1,2 \]

Each test problem was solved using the developed algorithm and three above mentioned algorithms. To overcome the problem of stochastic output of the simulation models, twenty replication of simulation model was considered and the test problems are solved by each method. The performance criteria were selected as the number of simulated points and the quality of final results.

Table (1) summarizes the computational results over 20 replications for the four considered algorithms in term of the best result. As it is seen in eight out of ten test problems the developed hybrid algorithm provided the optimal solutions. For two other problems, problems #4 and #5, the objective function values are very closed to the optimal, less than 0.1 and 0.3 percents.

Table (2) summarizes the computational results over 20 replications for the four considered algorithms in term of the average best result. As it is seen in four out of ten test problems the developed hybrid algorithm provided the average solutions equal to the optimal solutions. For the other problems, except problem #3 and #5 the average solutions have the deviations less than 1.1 percent from the optimal solutions. For problem #3 the dilation is 4.06 percent from the optimal solution. Only for problem #5 we obtained a solution point which is far away from the optimum which is caused to have an average solution with unacceptable deviation from the optimal solution.

Table (3) summarizes the number of simulated points for obtaining the final solution. As we emphasized earlier, it is assumed that the simulation runs are
expensive or from computation points of view they are assumed to be very time consuming. Therefore the number simulation points for obtaining the optimal solution is very critical performance measure in evaluation of the optimization algorithm. As it is seen considering this performance measure, the developed hybrid algorithm outperforms the other three algorithms. Actually in most of the ten test problems, the developed algorithm provided more than 80 percent saving in the number of simulated points comparing with the average number of points simulated by the other three algorithms.

Table 1: Best objective function value obtained comparing to other methods

<table>
<thead>
<tr>
<th>Problem No.</th>
<th>Optimum</th>
<th>NP</th>
<th>PSO</th>
<th>PSPO</th>
<th>Hybrid A.</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>-0.500</td>
<td>-0.493</td>
<td>-0.500</td>
<td>-0.500</td>
<td>-0.500</td>
</tr>
<tr>
<td>P2</td>
<td>-6.081</td>
<td>-6.079</td>
<td>-6.081</td>
<td>-6.081</td>
<td>-6.081</td>
</tr>
<tr>
<td>P4</td>
<td>-1.032</td>
<td>-1.032</td>
<td>-1.032</td>
<td>-1.032</td>
<td>-1.031</td>
</tr>
<tr>
<td>P5</td>
<td>0.000</td>
<td>0.006</td>
<td>0.001</td>
<td>0.000</td>
<td>0.003</td>
</tr>
<tr>
<td>P6</td>
<td>0.000</td>
<td>0.008</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>P7</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>P8</td>
<td>-1.801</td>
<td>-1.801</td>
<td>-1.801</td>
<td>-1.801</td>
<td>-1.801</td>
</tr>
<tr>
<td>P9</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>P10</td>
<td>0.398</td>
<td>0.398</td>
<td>0.398</td>
<td>0.398</td>
<td>0.398</td>
</tr>
</tbody>
</table>

Table 2: Average objective function value obtained comparing to other methods

<table>
<thead>
<tr>
<th>Problem No.</th>
<th>Optimum</th>
<th>Average objective function over replications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>NP</td>
</tr>
<tr>
<td>P1</td>
<td>-0.500</td>
<td>-0.344</td>
</tr>
<tr>
<td>P2</td>
<td>-6.081</td>
<td>-6.080</td>
</tr>
<tr>
<td>P4</td>
<td>-1.032</td>
<td>-1.031</td>
</tr>
<tr>
<td>P5</td>
<td>0.000</td>
<td>1.350</td>
</tr>
<tr>
<td>P6</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>P7</td>
<td>0.000</td>
<td>0.080</td>
</tr>
<tr>
<td>P8</td>
<td>-1.801</td>
<td>-1.733</td>
</tr>
<tr>
<td>P9</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>P10</td>
<td>0.398</td>
<td>0.498</td>
</tr>
</tbody>
</table>

Table 3: Number of simulated points to obtain the final solution

<table>
<thead>
<tr>
<th>Problem No.</th>
<th>Number of simulation points</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NP</td>
</tr>
<tr>
<td>P1</td>
<td>490</td>
</tr>
<tr>
<td>P2</td>
<td>570</td>
</tr>
<tr>
<td>P3</td>
<td>520</td>
</tr>
<tr>
<td>P4</td>
<td>810</td>
</tr>
<tr>
<td>P5</td>
<td>530</td>
</tr>
<tr>
<td>P6</td>
<td>610</td>
</tr>
<tr>
<td>P7</td>
<td>290</td>
</tr>
<tr>
<td>P8</td>
<td>390</td>
</tr>
<tr>
<td>P9</td>
<td>240</td>
</tr>
<tr>
<td>P10</td>
<td>340</td>
</tr>
</tbody>
</table>

V. Conclusions and Remarks

In this paper, a metamodel based hybrid algorithm was developed for optimization of the digital computer simulations models. It is assumed that the considered simulation models are computationally expensive, and have a single output function which is nonlinear continuous unconstrained function. By the computationally expensive we mean that, each simulation run is required an extensive computational time.

The hybrid algorithm is developed by modifying and integrating of several concepts and routines. We incorporated NP and PSO algorithms to develop an efficient search mechanism and we modified K metamodel to be applied in stochastic simulation-optimization model (SK), and is integrated to the search mechanism as a metamodel for facilitating the function fitting processes of the simulation’s output. As the result a hybrid algorithm is developed.
The efficiency of the developed hybrid algorithm was then evaluated through computational experiments. Six complex test problems were selected from the literatures and the efficiency of the developed hybrid algorithm was evaluated by comparing its performances against three other algorithms. Two algorithms were selected from the existing literatures, and one algorithm was a preliminary developed algorithm by the authors. The result of these computational experiments revealed that the developed hybrid algorithm can provide very robust solutions with a very low computational effort.

References Références Referencias

30. Reference to a book:
32. Reference to a chapter in an edited book: