OPTIMIZATION VIA SIMULATION TECHNIQUE USING A REVISED RSM

AZIMI, P[arham]

Abstract: The current study develops a new revised version of the so-called response surface methodology (RSM) by integrating RSM with mathematical programming and gradient methods. The main purpose of the research is to overcome common issues such as low convergence and time consuming process of defining/refining of sub-regions in comparison to the traditional versions of RSM. To test the performance of the algorithm, the application of it has been shown in simple and U-shaped assembly line balancing problems with stochastic processing times. Several experiments have been carried out to show the efficiency of the proposed algorithm using a set of large test problems taken from the literature.

Key words: RSM, optimization via simulation, assembly line balancing problem

1. INTRODUCTION

Realizing smooth mathematical formulations in optimizing real world applications would be too complex process which sometimes leads to simplified assumptions or developing heuristics. In such a case, it is prevalent to employ simulation to discover at least one admissible solution. Simulation-based Optimization or optimization via simulation is a commonly used technique with the aim of managing stochastic complex systems where the system performance can only be evaluated by simulation. Simulation Optimization offers a structured approach to stochastic complex system design when analytical expressions for input/output relationships are unavailable. Nowadays, Response Surface Methodology (RSM) is widely applied in Simulation Optimization. Applications of simulation are widely found in many areas including supply chain management, finance problems, manufacturing problems, engineering designs, medical treatments and building structures (Fu & Hu, 1995; Gürkan et al., 1994; Kim & Ding 2005; Semini & Fauske, 2006). Due to the complexity of a simulation model, the objective function is usually difficult and expensive to be evaluated. However, this is the strength point of simulation technique because it is more real (Azimi & Charmchi, 2011). Moreover, the inaccuracy of the objective function often complicates the optimization process because of the stochastic events in the real world systems. There are several excellent review papers on the subject of optimization via simulation, such as (Fu & Hu, 1995). The basic algorithm of RSM can be briefly described as follows:

Phase I) In this phase, the first order of experimental designs are used to get a least-squares fit, then a steepest descent direction is estimated from the model, and a new sub-region chosen to explore via:

$$\bar{x}_{n+1} = \bar{x}_n - a_n \nabla L_n \quad (1)$$

Where $\bar{x}_n$ is the representative point of the nth explored solution, ∇L_n is the estimated (from the fitted linear response) gradient direction, and a_n represents the step size determined by a linear search or some other means.

Phase II) A quadratic response surface is fitted using more detailed second order experimental designs, and then the optimal solution is determined analytically from this fit.

However, RSM has two main shortcomings. At first, RSM uses steepest ascent (or descent) method (SA) to search for the next improved region of interest (ROI). SA suffers from two main problems: (i) it is scale dependent; (ii) the step size along its path is selected intuitively. So, user involvements are required in any iteration of any steps. Secondly, RSM provides no convergence guarantee. In this paper, the outline of Simple and U-shaped ALB problem has been explained in section 2. The proposed optimization via simulation algorithm is presented in section 3. In section 4, computational results are explained and conclusions are provided in section 5.

2. SIMPLE AND U-SHAPED ALB PROBLEMS

The Simple Assembly Line Balancing (SALB) Problem aims to assign a set of tasks to a set of workstations such that:

A) Each task is assigned to only one workstation B) Precedence relations among the tasks are not violated and C) One or more objectives are optimized. The mathematical model of SALB-1 problem is:

i : index for the task number i , $\forall i = 1, 2, \dots, m$, j : index for the workstation j , $\forall j = 1, 2, \dots, n$, P : the set of precedence relationships among all tasks, E_i : the earliest workstation to which task i can be assigned according to the precedence relationships, L_i : the latest workstation to which task i can be assigned according to the precedence relationships, t_i : the time required by task i for completion, W_j : refers to the subset of all tasks that can be assigned to workstation j , $\|W_j\|$: refers to the number of tasks in subset W_j , P_i : refers to the set of tasks that must proceed task i , S_i : refers to the set of tasks that must succeed task i ,

$$z_j = \begin{cases} 1 & \text{if workstation } j \text{ is used} \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

$$x_{ij} = \begin{cases} 1 & \text{if task } i \text{ assigns to workstation } j \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

Problem 1:

$$\text{Min. } Z = \sum_{j=1}^n z_j \quad (4)$$

S.t.

$$\sum_{j=E_i}^{L_i} x_{ij} = 1 \quad \forall i = 1, 2, \dots, m \quad (5)$$

$$\sum_{i \in W_j} t_i x_{ij} \leq C \quad \forall j = 1, 2, \dots, n \quad (6)$$

$$\sum_{j=E_a}^{L_a} j x_{aj} \leq \sum_{j=E_b}^{L_b} j x_{bj} \quad \forall (a, b) \in P \quad (7)$$

$$\sum_{i \in W_j} x_{ij} \leq \|W_j\| z_j \quad \forall j = 1, 2, \dots, n \quad (8)$$

$$x_{ij}, z_j \in \{0, 1\} \quad \forall i = 1, 2, \dots, m \text{ and } \forall j = 1, 2, \dots, n \quad (9)$$

Where:

$$E_i = \left\lceil \frac{t_i + \sum_{k \in P_i} t_k}{C} \right\rceil \quad (10)$$

$$L_i = n + 1 - \left\lceil \frac{t_i + \sum_{k \in S_i} t_k}{C} \right\rceil^+ \quad (11)$$

Problem 1 is a NP-Hard problem (Scholl and Christian 2006). U-type (also called U-line or U-shaped) assembly line balancing is a generalization version of SALB problem which has great influence on system performance (Fig. 2). Its mathematical model is as the same as the simple one.

Fig 1. The U-shaped SALB problem.

3. PROPOSED ALGORITHM

The proposed algorithm has been designed for all combinatorial optimization problems where the objective function is explicitly known or unknown. The proposed algorithm is based on RSM and has four main phases such as: Phase 1) modeling the problem using simulation software and taking necessary samples from it. Phase 2) developing sufficient interpolations from different orders such as linear, second ordered,... interpolations for the objective function based on a predefined set of samples taken from the corresponding simulation model. Phase 3) using a linear continuous mathematical programming to obtain the starting point of the optimization process (here we relax all integer variables in Problem 1 to be in $[0,1]$ space). Phase 4) using the higher-ordered interpolations for computing the slope of the objective function to improve the current solution using the gradient method until having no further improvements. The best criterion to be measured for the accuracy of interpolations is correlation coefficient (r^2). When the value of r^2 is equal or less than the desired one, the phase terminates and the corresponding interpolation is used in the next phases. For calculating the position of the next solution, equation (1) is used. The step size (a_t) is obtained from the following 0-1 programming model to prevent generating any infeasible solutions for the next movement:

Problem 2

$$\text{Max. } Z = a_t \quad (12)$$

S.t.

$$\text{All constraints in Problem 2} \quad (13)$$

$$\tilde{x}_{t+1} = \tilde{x}_t - a_t \nabla F(\tilde{x}_t) \quad (14)$$

$$x_{ij} \in \{0,1\} \text{ and } a_t \geq 0 \quad (15)$$

The proposed algorithm, avoids the common issues in traditional RSM such as (i) no need to calculate the slope of the objective function by generating several replications at each movement (ii) the starting point is selected intelligently not a random blind point (iii) the algorithm does not depend on any sub-procedures for defining of sub-regions in the feasible area and refining them through the algorithm. These sub-processes are usually time consuming processes with several statistical calculations.

4. COMPUTATIONAL RESULTS

All computations were run on a PC with 4 MBytes Ram and Core I5, 2.67 GHz CPU. Enterprise Dynamics (ED) 8.1 was used as the simulation software, Lingo 8.0 to solve the mathematical models and Visual Basic 2007 as coordinator

between Lingo and ED. In order to test the performance of the proposed algorithm, a set of 168 test problems was taken from the literature and the results were compared to (Baykasoğlu & Dereli, 2009) and listed in Table 1. As the results show, the performance of the proposed algorithm dominates the competitor in terms of speed even for large scale problems. On average, the computational time for the simulation-based algorithm was 5.95 Sec. for simple and 13.98 Sec. for U-shaped models while the competitor algorithm consumed 39.51 and 153.30 seconds for simple and U-shaped models, respectively. The algorithm was better or equal than that of its competitor in 161 cases out of 168 which shows its strength in applications.

	Bayk.'s simple result for Obj. Function	Simulation simple results for Obj. Function	Bayk.'s U-shaped result for Obj. Function	Simulation U- shaped results for Obj. Function	Bayk.'s simple Comp. time	Simulation simple Comp. time	Bayk.'s U-shaped Comp. time	Simulation simple Comp. time	No. of cases that Simulation dominated or
Ave. values	25.55	25.33	25.22	25.24	27.93	39.51	153.30	13.98	161

Tab. 1. Experimental Results

5. CONCLUSIONS

In this research, a new version of Response Surface Method has been developed by integrating mathematical programming and gradient method with RSM. This new version of RSM has not common issues in traditional versions such as low convergence, need for several replications of the simulation model and defining/refining sub-regions to find the near optimum solutions. Finally, it was compared to a similar study in the literature. Regarding the research limitations, if the base computer had more powerful CPU and the new version of the simulation software was used, the results could be better. For future works, it is highly recommended to integrate other metaheuristic algorithms such as Particle Swarm Optimization or Ant colony optimization with simulation to obtain better results.

6. REFERENCES

- Azimi, P. & Charmchi, H.R. (2011). A new heuristic algorithm for the dynamic facility layout problem with budget constraint. *Proceedings of International Conference on Modeling and Simulation, ICMS 2011*, the Netherlands, 2010-376X, pp. 1181-1187, WASET, Amsterdam.
- Baykasoğlu, A. & Dereli, T. (2009). Simple and U-type assembly line balancing using an ant colony based algorithm. *Mathematical and Computational Applications*, 14, 1-12.
- Fu, M.C. & Hu, J. (1995). Sensitivity analysis for Mont Carlo simulation for option pricing. *Probability in Engineering and Information Science*, 9(3), 417-446.
- Gürkan, G.; Özge, A.Y. & Robinson, S.M. (1994). Simple-path optimization in simulation. *Proceedings of the 1994 Winter Simulation Conference*, USA, pp. 247-257, 9780321081, Society for Computer Simulation Internal, Florida.
- Kim, P. & Ding, Y. (2005). Optional engineering system design guided by data-mining methods. *Technometrics*, 47(3), 336-348.
- Scholl, A. & Christian, B. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing problem. *European Journal of Operations Research*, 168, No.3, 666-693.
- Semini, M. & Fauske, H. (2006). Applications of discrete-event simulation to support manufacture logistics decision-making: A survey. *Proceedings of the 2006 Winter Simulation Conference*, WSC '06, 1-4244-0501-7, pp. 1946-1953, Society for Computer Simulation Internal, USA.