Journal of Naval Science and Engineering
2014, Vol.10, No.1, pp.13-31

MULTI META-HEURISTICS FOR SIMULATION OPTIMISATION

Serdar BOZOĞLAN [1]
Murat M.GÜNAL[2]

[1,2]Industrial Engineering Department,
Turkish Naval Academy, Naval Sciences and Engineering Institute, Tuzla, Istanbul
[1] zserdarb@hotmail.com, [2]mgunal@dho.edu.tr

Abstract

Optimisation with simulation is a happy marriage of two Operations Research methods. In the last decade, the research in this field has accelerated and many researchers have been interested in Simulation Optimization (SO). New techniques have been developed as a result of this interest. Almost all commercial simulation software contains an optimization module. Generally, these modules exploit meta-heuristic methods; however, they do not allow the analyst to choose the method. The performance of meta-heuristic methods may depend on the problem type and therefore the choice of method is crucial. In this paper, we aim to fill this gap and presented an open-source java-based simulation-optimization code library. The library includes three heuristic methods; genetic algorithm, tabu search, simulated annealing, as well as three enumeration based methods; partial and complete enumeration, and a new neighbourhood-based heuristic method. At the simulation side, Simkit, an event-based and open-source simulation library, is used. At the application side, we defined a fictional optimisation problem and used it to compare performances of the algorithms. Our results demonstrated the potential benefits of having multi meta-heuristics available in SO.

BENZETİMLE ENİYİLEME İÇİN ÇOKLU META-SEZGİSELLER

Özetçe

Benzetim ile eniyileme iki yöneylem araştırması yönteminin mutlu bir evliliğidir. Son on yılda bu alandaki araştırmalar ivme kazanmış ve birçok araştırmacı Benzetimle Eniyileme (BE) alanına ilgi göstermiştir. Bu ilginin sonucu olarak yeni yöntemler de geliştirilmiştir. Hemen hemen bütün ticari benzetim yazılımları bir çeşit BE modülü içermektedir. Genel olarak bu modüller meta-sezgisel yöntemleri kullanmaktadır ancak analizcinin yöntem seçimine izin vermemektedir. Meta sezgisel yöntemlerin problem tipine bağlı olarak performansları değişebilir ve bu nedenle de yöntem seçimi önemlidir. Bu makalede bu açığı doldurmayı hedefliyoruz ve açık kaynak kodlu Java tabanlı bir BE kod kütüphanesi sunuyoruz. Kütüphane üç meta sezgisel; Genetik algoritma, yasaklı arama, simulated annealing, ve üç sıralı aramalı algoritma; parçalı ve tam sıralı aramalı, ve yeni bir komşuluk tabanlı sezgisel yöntemi içermektedir. Benzetim tarafında ise açık kaynak kodlu ve olay tabanlı bir kütüphane olan Simkit kullanılmıştır. Uygulama olarak hayali bir

eniyileme problemi tanımlanmış ve algoritmalar karşılaştırılmıştır. Çalışmanın sonuçları BE'de çoklu meta sezgisellere sahip olmanın potansiyel faydalarını göstermiştir.

Keywords: *Simulation Optimization, heuristics, genetic algorithm, tabu search.*
Anahtar Kelimeler: *Benzetim, Eniyileme, Sezgisel yöntemler, yasaklı arama, genetic algoritma.*

1. INTRODUCTION

In most of today's simulation software in the market, optimization modules are included in one way or another. Some includes third party optimization bundles, and some includes embedded algorithms to optimize the parameter values of a simulation model. Particularly, when a solution to a problem is analytically not tractable, simulation can be used as a tool to model the problem in hand. In Simulation Optimization (SO), there is a simulation model which is repeatedly run to explore how the outputs change by different input values. As in traditional optimization problems, there is an objective function and constraints, and the aim is to find the best input values which maximizes (or minimizes) the objective function value by satisfying the constraints.

There are six commonly used methods in SO (Fu et al (2005)). The first one is "ranking and selection" method. In this method, a list of available solutions and their Objective Function Value (OFV)s are created and the best is chosen (the minimum or the maximum). This method is useful when there is a fix set of alternative solutions. It is important to note that "an available solution" means a set of input values of the simulation model, and an OFV means an output value of the simulation model.

Second method is the Response Surface Methodology (RSM). Its origins are in statistical design of experiments and eventually its task is to seek for the relationship between the inputs (factors) and outputs (response) of the model. After examining the relationship, a meta-model is built and then deterministic optimization methods are applied to find an optimum solution. The third method is gradient based procedure which eventually mimics the methodology in RSM. This method looks for the movements in

gradient directions where the changes are significant in the output. It requires differential equations calculations since the gradients are the changes in the slope in response curves. However this method is known to be better performed when the inputs are continuous variables. The fourth method is random search and works as in the gradient search. Although the search is random, the method proceeds systematically and iteratively where a neighborhood structure is involved. This method can be applied in model with both discrete and continuous input variables. The neighborhood is significant since a candidate solution may not be feasible. The fifth method in SO is Sample Path Optimization. The method is based on deterministic optimization methods (e.g. linear programming) on estimates of n simulation replications. And finally, the sixth method is the use of Meta-heuristics. Meta-heuristics are search strategies in solution space in order not to trap to local optimums. It is perhaps the most popular SO method in today's SO community.

In this paper, we explore feasibility of use of multiple meta-heuristics in SO. The motivation of our research comes from the literature review presented in the following section. Our review revealed that most simulation software in the market use meta-heuristic search methods for SO, however, in these software, interestingly, only one or two meta-heuristics are embedded and the user is not free to choose the search method. The choice of method is left to the software. These two issues observed in simulation software are the driving forces of our work. What benefits can be gained by having multiple heuristics available to the user and the freedom of selecting one of these methods is explored.

This work is intended as a proof of concept in SO. The concept is to let users to choose a method from multiple meta-heuristics available. To demonstrate the concept in reality, a software library (HePSi (Heuristics Package for Simulations)) which is developed by the authors is presented. We used the library in an imaginary test problem that we inspired from a well-known optimization problem, Travelling Salesman Problem (TSP) applied in maritime transportation.

2. BACKGROUND LITERATURE

The literature in this area does not date back to 1990s, since the two merging topics grown on their own for many years. Later in 80s, simulation and optimization have been seen together. A good starting point is the literature reviews such as Law and McComas (2000), Sabuncuoglu and Tekin (2004), and Fu et al (2005). Additionally, there has been progress in SO methodology, for example, Hong and Nelson (2006, 2007) introduce a new methodology for optimization via simulation. Their method is based on stochastic search formed by integer decision variables and guarantees to converge locally to an optimum. In the application side, SO is applied in many domains such as inventory systems (Alferaei and Diabat, 2009), project management (April et al 2004), and supply chains (Zhao and Sen, 2006). Willis and Jones (2008) use heuristics for multi objective SO. Their framework combines a simulation model with a non-exhaustive heuristic search algorithm with an embedded multi-objective optimization technique.

Heuristic search methods are popular due to their advantages over the other SO methods. Firstly, the simple black-box approach fits in heuristic methods; that is meta-heuristics optimizer generates a candidate solution and this candidate solution is supplied to a simulation model to obtain an OFV. The simulation model produces an output, based on the inputs generated by the meta-heuristics optimizer and the optimizer then generates a new input set and so on. This cycle goes on until a good solution is obtained. The generated solution must be a feasible solution. Secondly, the iterative nature of this approach is relatively simple to implement since the model and the optimizer work independently. Once a model is built, iteratively experimentation is conducted with it. Meta-heuristics methods such as Simulated Annealing (Willis and Jones 2008, Alferaei and Diabat 2009), Tabu Search (Dengiz and Alabas 2000), Genetic Algorithm (Homaifar et al 1994) and Neural Networks are among the popular methods.

There are views on the practice of SO, for example Fu (2002) indicates that the research on SO is disconnected in academia and in practice since there are, he argues, differences between both parties'

expectations. The academia searches the ways of "better convergence" to optimum, and the industry looks for the ways of practicalities of these methods. He also argues that commercial simulation optimization software has been successful because, firstly, there is market for them and they are sold with simulation modeling software. Secondly, these products handle complex problems and produce "good" results in a timely manner. This actually echoes Law (2007)'s views where a list of optimization modules in commercial simulation software is supplied (p. 660, Table 12.19). The list includes most commonly used simulation products and the optimization software supplied with them, e.g. AutoStat, OptQuest, OPTIMIZ, SimRunner, and WITNESS Optimizer, and the algorithms and search strategies embedded in these software, e.g. evolution strategies, scatter search, tabu search, neural networks, genetic algorithms, simulated annealing etc. This survey reveals that in commercial software, heuristic search methods are most popular. Another point it reveals is that one or two (or exceptionally in OptQuest; Tabu Search, Neural Networks, and Scatter Search are combined into a single search heuristic) meth-ods are implemented in these software. Additionally, the software automatically chooses the method and the users are not allowed to select a method

3. CONCEPTUAL REPRESENTATION OF HEPSI

HePSi (Heuristics Package for Simulations) is a meta-heuristics code library written in Java for SO. The package consists of Genetic, Simulated Annealing and Tabu Search meta-heuristics, partial and complete enumeration algorithms, and a new and improved heuristic of neighborhood-based partial enumeration algorithm (Figure 1).

Since the scope of HePSi is not to create full generic software, there are limitations:

- HePSi is designed for discrete-event simulations (DES) built in Simkit (Buss, 2010), a Java based DES software library.
- It works independent from the simulation model. A model only interacts with the package through input variables.

- Discrete decision variables are allowed; that is a solution in the optimization problem can only have integer values, and bounds of these variables can only be integers (e.g. the cities visited in a TSP route, the number of machines in a job shop, the number of nurses in a staff roster).

Although these limitations exist, HePSi is intended to be a generic package. It can fit in any optimization problem that can be solved using the meta-heuristic algorithms mentioned above with minor modifications in the package. It is specifically designed to run independently with a simulation model. An executable class links the two, HePSi and the simulation model, and act as a communicator between them. Parameters of the meta-heuristics algorithms and the inputs of simulation model are entered in the execution class. In the following sections, specifics of the modules in HePSi are presented.

Partial Enumeration (PE)

This is the simplest and primitive part of HePSi. This class is especially needed when the solution space is extremely large and scanning the whole solution space requires too much time. Note that reducing the number of variables also reduce the size of the solution space (Law, 2007) which significantly improves the performance. The search is conducted randomly in PE. It starts from a random solution and travels randomly in the solution space while checking the feasibility of the solution. It never guarantees any optimal solution, but gives an indication of the space. It helps the user to evaluate how decision variables affect the objective function value. There is one parameter of this algorithm; the proportion of the search space, e.g 10% of all possible and feasible solutions, that is to be evaluated. Obviously, this module does not guarantee an optimum, but is useful if the search space is large. PE can be used in factorial design experimentation.

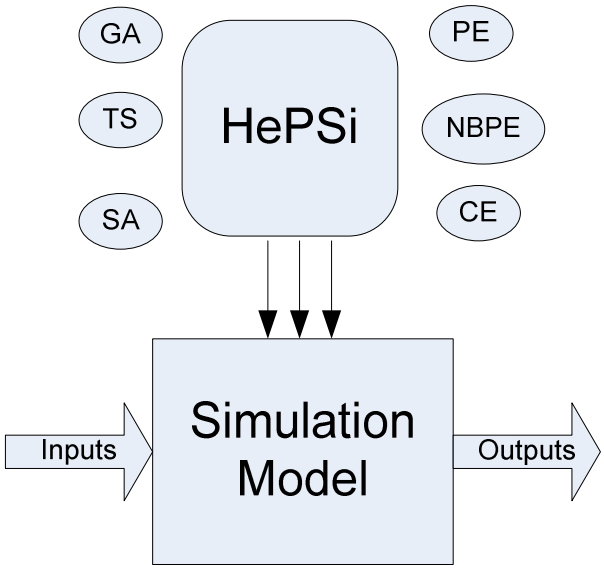

Figure 1. Conceptual Representation of HePSi.

Neighborhood-Based Partial Enumeration (NBPE)

This method is similar to PE. The search is done randomly but more systematically. First a number of points in the search space is entered and the algorithm proceeds with local searches in these points.

Complete Enumeration (CE)

The efficiency of this module depends on the size of the solution space, since in this module the whole solution space is scanned. CE guarantees the optimum solution. However, searching the whole solution space requires too much run time, and is in fact impossible when the size of the problem is large. Even the problem size is reasonable, this module can be beaten by the other algorithms. However since it guarantees the optimum solution, it can be used for benchmarking (e.g. simulated annealing finds a

near optimum solution in x seconds, and CE finds the optimum in x + or - y seconds).

Genetic Algorithm (GA)

GA is very popular as an optimization technique due to its generality. Its concepts are influenced from evolution theory. It searches the solution space iteratively and best solutions are evolved and others are discarded.

In HePSi's GA module, GA parameters are chosen by the user before the simulation model runs. These parameters are population size, crossover and mutation rates, the policy to select the crossover point, and the policy to select the mutation point. The GA operations may need some adaptations to the problem domain. HePSi's GA module is designed to allow these adaptations. Consider a TSP where a complete tour is a solution. This means that a city is not to be revisited. In GA's crossover terms, two solutions may have the same sequence of cities in which the crossover operation may result in solutions with revisited cities. The mutation operation has also similar attributes, e.g. a city to mutate violates the rule of unvisited city. In the GA module, this kind of problem specific constraints can be coded.

Simulated Annealing (SA)

This heuristic algorithm is based on physical activity of annealing of metals. The algorithm in-spires the atomic structure of solid objects such as metals. Metals get their forms perfectly while they are cooling or loosing temperature. This process continues until the metal crystallizes completely. This algorithm also works iteratively. There are two notions in SA; the neighborhood of solutions, and the temperature. SA algorithm depends on one of the random neighbors of current solution in an iteration.

As in the GA module, the parameters of SA algorithm (the probability of accepting a "bad" solution) is entered before the simulation

run. There are problem domain specifics in SA, such as the definition of the neighborhood. These specifics can be amended in the code to fit SA to the problem.

Tabu Search (TS)

This method explores solution space beyond local optimality. Local search is based on exploring neighborhood of any given candidate solution. The best solution in a neighborhood is chosen even if it causes deterioration, worse than the current candidate solution. This strategy enables to avoid trapping in local optima. The algorithm memorizes old candidate solutions and does not al-low progress in old candidate solutions' direction for a while. That means algorithm imposes a tabu. When a neighbor of a candidate solution is chosen as a current solution, the change is considered as tabu. While a defined iteration size, this change is not considered as swap, except this change enables being the best solution ever. Tabu list is composed of recently chosen candidate solutions. It prohibits choosing better solutions as current solution to avoid local optima.

TS algorithm is implemented in HePSi where its parameters are entered in the execution module. As in the other algorithms, the user can adopt domain specific features in the source code.

4. A TEST CASE FOR HEPSI

Problem definition

We used a well-known optimization problem to test HepSi. There is an imaginary company which desires to define the best route for its one container ship to maximize its profit. This test problem is a combination of "Travelling Salesman Problem (TSP)" and the knapsack problem. We assume that there are several ports that are to be visited once in a tour. An example tour map can be seen in Figure 2.

The company declares direct transportation charges of one unit cargo. The direct transportation charge matrix is diagonally symmetric and is a user input. These tables show the charge of transportation of one unit cargo directly from one port to another. However in the ship's route a port can be visited after visiting some other ports and therefore the charge may vary depending on the ports visited previously. The rationale is that when the ship takes a cargo directly to any port, delivery occurs in shorter time however when the ship takes a cargo indirectly, the cargo owner must wait longer for its cargo. Therefore cargo fees depend on the distance and the number of ports visited. Ship owner company tries not to lose any customer and therefore decreases transportation fees for the late delivery when the ship goes indirectly to any port. We call this "corrected charge" of transportation

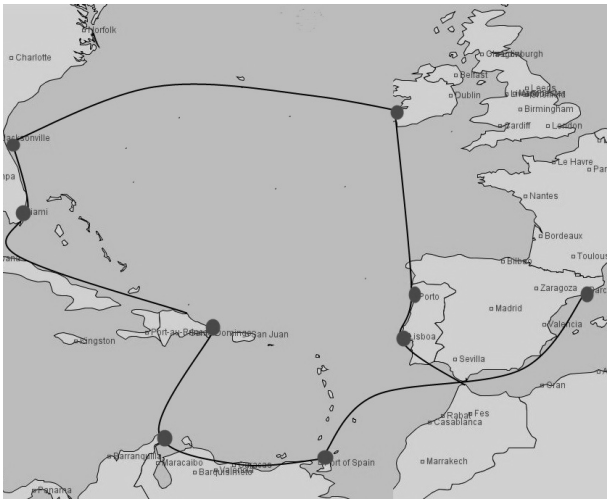

Fig. 2. A sample tour map.

The demand for transportation is stochastic which makes this problem appealing for a DES model. The number of containers that is to be transferred to a port is determined from a stationary distribution. The ship cannot know the quantity of next port's cargo in advance. For example, when the ship is in port 4 and the route is [0,3,10,5,4,6,7,9,8,2,0], the ship is

incapable of knowing the cargos from port 6 to anywhere or from port 7 to anywhere etc. We assume that the demand (number of containers) that needs to be transferred to another port is normally distributed.

Three basic scenarios are determined. In the first scenario, there are 9 ports, in the second scenario the port number is increased to 12, and in the last scenario, port size is extended to 15. In these scenarios, initial port is always port 0. Additionally, each scenario is also categorized into two main sub scenarios by altering the carriage capacity of the ship. The distances are given in a matrix and measured in nautical miles. The details of each scenario and assumption of the problems are given below.

The model

The model is implemented in Simkit (Buss, 2010). Simkit is a Java based DES library which implements event-based modelling. Before writing code in Simkit, the modeller builds an Event Graph (EG) of the system to be modelled. Event graphing is a very efficient method for representing events and their interactions in a system. An EG has two elements; nodes (events) and edges (event transitions).

The EG of the system described in the previous section is simple and includes two events; an arrival event and a departure event. When the ship arrives to a port, it gets the cargo and related statistics are updated. An arrival event also schedules a departure event to the travel time between ports. The two events are executed iteratively until all ports are visited. Note that the route is determined before the simulation run by the optimizer.

Problem Specifics in HePSi

As discussed earlier, HePSi is intended for generic use but some customizations are necessary. For example in the problem defined above, a feasible solution is represented by an array of integers which indicates port numbers. In this array, though, every integer must exist only once since every port is to be visited once.

Adaptations in heuristic algorithms are also required. In GA module, every time a crossover and mutation operation is executed, the new solution (mutated or crossovered) must be checked for its feasibility. For example, mutation by one port is not possible for feasibility and therefore displacement mutation is appropriate (Michalewicz (1992)). In SA module, definition of neighbourhood is important since this algorithm progresses towards neighbours. A neighbour of a port in our problem is chosen as the highest direct charge per distance port. If the first neighbour is already in the solution set (the route) then second neighbour is included and so on.

Experimentation

We defined three main scenarios, each having two sub-scenarios, as presented in Table 1. In main scenarios, we altered the number of ports that the container ship visits. Sub-scenarios include the capacity of the ship. Container quantities are measured in TEU (Twenty-foot Equivalent Unit) which is a standard measure in maritime transportation. Partial Enumeration (PE), Neighborhood-Based Heuristic (NBPE), Genetic Algorithm (GA), Tabu Search (TS) and Simulated Annealing (SA) methods of HePSi are applied to all scenarios. We conducted our experiments on a computer with 2 Gb RAM and 2.2 GHz Intel Core 2 Duo Processor.

Table 1. Configuration of scenarios

Scenario No (Main-Sub)	Number of ports (incl.Port 0)	Cargo ship capacity (TEU)
1-1	9	400
1-2	9	4000
2-1	12	400
2-2	12	4000
3-1	15	400
3-2	15	4000

Scenario 1

There 9 ports in this scenario and therefore the distance matrix and direct charge matrix are 9 by 9. The demand, number of containers which require transportation at ports, are distributed normally. The mean values of number of cargos are user inputs and given as a matrix. Coefficient of variation (cv) is 20 percent of mean value for each port. For instance, the mean value of the cargo from port1 to port4 is 60. Thus, cv is 20% of mean value of 60 TEU which is 12 TEU.

We evaluated that 60 replications are enough for estimating outputs of scenario-1. In this scenario, approximately 50% of 40,320 solutions are explored.

Scenario 2

This scenario includes 12 ports which increased the number of feasible solutions to 39,916,800. As expected, the curse of dimensionality in optimization problems affected the run time and the area to scan. In this scenario, approximately 0.2% of whole solution space is explored and 70 replications are needed.

Scenario 3

This scenario includes 15 ports which caused the number of solutions to increase to 87,178,291,200. Only 1/750,000 of whole solution space is explored to find the optima. We evaluated that 85 replications are enough for this scenario.

Results of the experiments are shown in Table 2. Note that this table gives the best result of each experiment. For example for Scenario 1-1, the best solution, that is the maximum Objective Function Value (OFV) of $151,910, is found when Tabu Search (TS) algorithm is applied. However, this table also shows that it took 5.8 minutes to achieve this solution and by scanning 22% of the solution space. Since feasible solutions are generated

based on a mechanism in that algorithm, it does not guarantee that the solution is not generated before. A history is kept to check that solutions are not fed to simulation model more than once.

5. DISCUSSION AND CONCLUSIONS

Simulation optimization (SO) is an active research area both in academia and in simulation software industry. Almost all commercial simulation software has SO modules, and almost all of them use meta-heuristic techniques for searching the optima. However, very few use multiple heuristics and none, to the best of our knowledge, of these software tools allow users to choose a method. Our work aims at contributing to the discussion in this area and to evaluate the potential of having multiple meta-heuristics. To achieve this objective, we developed a computer code library, Heuristic Package for Simulations (HePSi), which implements well-known meta-heuristic algorithms. HePSi is implemented in Java and can be used attached to Simkit, a discrete event simulation (DES) library. The heuristics and meta-heuristic methods included in HePSi are genetic algorithm, tabu search, simulated annealing, neighborhood-based partial enumeration heuristic, partial and complete enumeration.

To test our approach, we created an imaginary maritime transportation problem where a shipping company wants to determine his best profitable ship route. The demand for transportation in this problem is stochastic. First, a simulation model of this imaginary system is built using Simkit and an optimum route is sought using HePSi. As the objective of the code library is to allow comparison, each heuristic method is applied to the problem. This extensive experimentation yielded 6 tables, a brief of them is given in Table 2. A table of this kind can help analyst to compare outputs of different optimization methods and therefore give a great flexibility which is not presented by other commercial SO software in the market.

The use of HePSi is independent from the modeling software. The modeler builds a model and later HePSi is used to optimize the problem. At

the model side, to demonstrate this approach, Simkit is used. Our preference is due to its power in constructing event-based flexible simulation models.

In addition to meta-heuristics implemented in HePSi, we propose a new heuristic, Neighbor-hood-Based Partial Enumaration (NBPE). The analyst has some control on the randomness of the search on the solution space by NBPE. This limited control distinguishes the algorithm partial enumeration. To speed up the search for optimum, we propose history structure for Genetic Algorithm, Tabu Search and Simulated Annealing. This structure lessens the run time considerably especially when the same solutions are generated. When a non-existing solution is produced, it is pushed into the history with the objective function value. If the same solution is reproduced, the simulation model is not run; the objective function value is taken from the history. Therefore the computation cost of the algorithms decreases considerably.

Experimentation for the test problem showed that parameter values of the algorithms are determinant factors of the solutions. For example by increasing the mutation rate in GA, or by increasing the temperature coefficient value in SA, we can get better results in shorter time. Seeing this kind of interactions in the experiments is a clear benefit of HePSi.

The parameter values in meta-heuristics affect the efficiency of the algorithms significantly. In this study, parameter tuning is done manually. That is after trying different values the best known parameter values are chosen. Therefore choosing the right values for the algorithms is a limitation of the study.

More applications are needed to justify the generality of the code library. In this context, more problems are intended to be solved using HePSi. This will increase the robustness as well as the generality. Additionally, the future study may include a distributed version of HePSi where a problem can be divided into sub-problems, or the methods in HePSi can be distributed to different processors. This results in obtaining and

comparing the results of the different algorithms in shorter time and therefore more time can be dedicated to experimentation.

Table 2. Comparative Results of the Scenarios

Scenario	Method	Route	OFV ($) Mean ± 95% C.I.	Run time (minutes)	Intended Scanned - %	Actual Scanned - %	Method Specific Parameters
1-1	TS	[0, 7, 5, 6, 3, 1, 4, 2, 8, 0]	151910 ± 3324	5.8	20132 --- 49.9%	9242 – 22.9%	Tabu Iteration: 720 Tabu count: 7
1-2	PE	[0, 7, 5, 4, 3, 1, 6, 8, 2, 0]	308812 ± 5025	68.3	20160 --- 50.0%	20160 --- 50.0%	
2-1	NBPE	[0, 8, 11, 3, 4, 5, 6, 1, 7, 2, 10, 9, 0]	204700 ± 3274	30.4	79590 - 0.20%	79340 - 0.20%	# of Iteration: 15 --- maxRandom: 4 factorial coefficient: 0.45
2-2	NBPE	[0, 10, 11, 2, 1, 3, 6, 5, 9, 7, 4, 8, 0]	447504 ± 4645	32.5	79590 - 0.20%	79590 - 0.20%	# of Iteration: 15 --- maxRandom: 4 factorial coefficient: 0.45
3-1	SA	[0, 11, 10, 3, 2, 7, 14, 1, 4, 5, 10, 6, 9, 8, 12, 0]	232340 ± 3734	52.4	116250 --- 1.3*(10^-4)%	51340 --- 9.5*(10^-5)%	# of Outer iteration: 7750 # of Inner iteration: 15 Temperature: 0.4 Primary Neighbor Size:7
3-2	GA	[0, 7, 11, 4, 1, 3, 14, 10, 5, 6, 8, 9, 12, 13, 2, 0]	530182 ± 5220	36.2	116205 --- 1.3*(10^-4)%	38960 --- 5.9*(10^-5)%	Crossover rate: 0.5 --- Mutation rate: 0.2 Population size: 4 x port size

REFERENCES

[1] J. April, M. Better, F.Glover, P.J.Kelly (2004) New advances and applications for marrying simulation and optimization. In Proceedings of the Winter Simulation Conference, R .G. In-galls et al., Eds. IEEE, Piscataway, NJ. 80–86.

[2] M.H.Alferaei, A.H.Diabat (2009) A Simulated Annealing Technique for multi-objective simulation optimization. Applied mathematics and computation.2009 215 pp 3029-3035

[3] F.Azadivar (1999) Simulation optimization methodologies. Proceedings of the 1999 Winter Simulation Conference.

[4] A.H.Buss (2001) Basic Event Graph Modeling. Technical Notes, Simulation News Europe, April 2001: 1-6.

[5] B.Dengiz, C. Alabas (2000) Simulation Optimization Using Tabu Search. Proceedings of the 2000 Winter Simulation Conference. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

[6] M.C. Fu (2002) Optimization for simulation: Theory vs. practice. INFORMS Journal on Computing 14:192-215.

[7] M.C. Fu, F.W. Glover, J.April (2005) Simulation Optimization: A Review, New Develop-mens, and Applications. Proceedings of the 2005 Winter Simulation Conference. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

[8] A.Homaifar, C. X. Qi, S.H. Lai (1994) Constrained optimization via genetic algorithms. SIMULATION 62(4), 242–253.

[9] L. J. Hong, B.L. Nelson (2006) Discrete optimization via simulation using COMPASS. Operations Research 54:115-129.

[10] L. J.Hong, B.L. Nelson (2007) A framework for locally convergent random-search algorithms for discrete optimization via simulation. ACM Transactions on Modeling and Computer Simulation (TOMACS) 17(4), 19.

[11] A.Law, M.McComas (2000) Simulation-based optimization. Proceedings of the 2000 Winter Simulation Conference.

[12] A.M.Law (2007) Simulation Modeling and Analysis, 4th ed. McGraw-Hill, New York.

[13] Z.Michalewicz (1992) Genetic Algoritms + Data Structures = Evolution Programs, Springer Verlag, Berlin Hiedelberg.

[14] G. H.Neddermeijer, G.J. van Oortmarssen, N. Piersma, R. Dekker (2000) A framework for response surface methodology for simulation optimization. In WSC ’00: Proceedings of the 32nd conference on Winter simulation, San Diego, CA, USA, pp. 129–136. Society for Computer Simulation International.
[15] İ.Sabuncuoglu, E. Tekin (2004) Simulation Optimization: A Comprehensive Review on Theory and Applications, IIE Transactions, Vol: 36, pp: 1067-1081, 2004.
[16] K.O.Willis, D.F. Jones (2008) Multi-objective simulation optimization through search heuristics and relational database analysis. Decision Support Systems 46 (2008) 277–286.
[17] L.Zhao, S. Sen (2006) A comparison of sample-path-based simulation-optimization and stochastic decomposition for multi-location transshipment problems, Proceedings of the 37th conference on Winter simulation, December 03-06, 2006, Monterey, California.