computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific Journal of Computing

OPTIMIZATION VIA SIMULATION: SOLUTION CONCEPTS, ALGORITHMS, PARALLEL COMPUTING STRATEGIES AND COMMERCIAL SOFTWARE

Gianpaolo Ghiani[1], Pasquale Legato[2], Roberto Musmanno[2], Francesca Vocaturo[2]

1) Dipartimento di Ingegneria dell'Innovazione Università degli Studi di Lecce, 73100 Lecce, Italy gianpaolo.ghiani@unile.it
2) Dipartimento di Elettronica, Informatica e Sistemistica Università della Calabria, 87030 Rende (CS), Italy {legato,musmanno, vocaturo}@unical.it

Abstract. *Simulation optimization (or optimization via simulation) is defined as the optimization of performance measures based on outputs from stochastic simulations. Although several articles on this topic have been published, the literature on optimization via simulation is still in its infancy. In this paper the research in this field is reviewed and some issues that have not received attention so far are highlighted. In particular, a survey of solution methodologies is presented, followed by a critical review of parallel computing strategies and commercial software packages. A particular emphasis is put on problems with discrete decision variables.*

Keywords: *Stochastic Systems, Simulation Optimization, Metaheuristics, Parallelization Strategy*

1. INTRODUCTION

Several problems arising in complex system design and operations are characterized by a inherent stochastic nature. While analytical models are viable under relative easy assumptions, their use becomes more and more difficult as the complexity of the system increases. In such situations, we resort to *simulation optimization* modeling as an effective alternative. Simulation optimization (or *optimization via simulation*) is defined as the optimization of the performance measures of complex systems based on outputs from stochastic (primarily discrete-event) simulations (see Figure 1).

More formally, the class of problems we are concerned amounts to finding a solution θ such that:

$$\min_{\theta\in\Theta} f(\theta) \qquad (1)$$

where $f(\theta)=E[L(\theta,\omega)]$, L is the sample performance measure, $\theta\in\Theta$ represents the (vector of) input variables (or controllable parameter settings), and ω represents a sample path (simulation replication). A particular setting of the variables is usually called either a *configuration* or a *design* while outputs are named *performance measures*, *criteria*, or *responses*. The constraint set Θ may be either explicitly given or implicitly defined. For simplicity in exposition, we assume throughout that the minimum exists and is finite, e.g., Θ is compact.

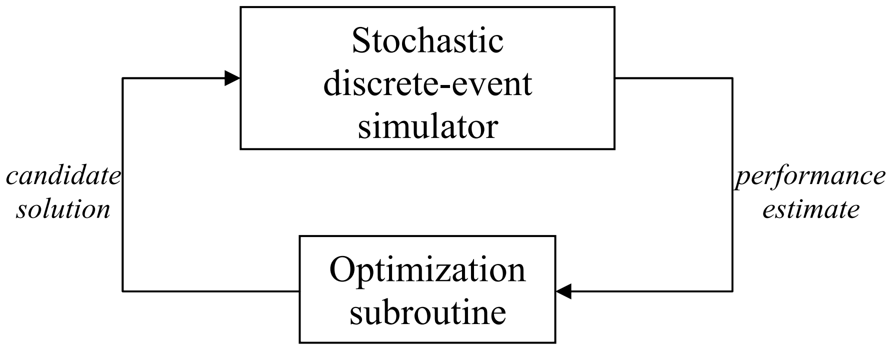

Fig. 1 - Simulation-Optimization approach

The peculiar features of this class of problems set up a dichotomy not present in deterministic optimization: that of the search process versus the evaluation process. In other words, simulation optimization involves two important parts: generating candidate solutions and estimating their objective function value. Hence, the actual *process* of optimization can be divided into two parts: (i) generating candidate solutions; (ii) evaluating such solutions. In optimization simulation, most of the computation is expended in phase (ii), a reversal of the deterministic setting, where the search (i) is the primary computational burden.

A major determinant of the computational cost for a simulation optimization algorithm is the number of simulation replications used to estimate J(θ) for each θ. There is no reason a priori to assume that the number of replications should be the same for all values of θ nor the same for each iteration in the search process. In sum, a key feature that is not a factor in deterministic settings is the trade-off between the amount of computational effort needed to estimate the performance at a particular solution versus the effort in finding improved solution points. Since evaluating and comparing are considered essentially the same step in the deterministic setting, currently implemented simulation optimization routines do not really address the notion of ordinal comparisons.

Simulation optimization techniques can be classified based on the nature of the feasible region. If it is a continuous set, then it may be appropriate to use a gradient based search method (see Andradóttir, 1998, and Fu, 1994, 2002, for surveys). If it is finite and fairly small, then it is possible to use suitable statistics methods, whereas, if it is finite but combinatorially large, it is impracticable to evaluate every alternative and a metaheuristic may be appropriate. In the following, the discrete case is examined in greater detail since it is much more relevant in engineering applications (e.g., in the design of computer systems, communication networks, flexible manufacturing systems, production assembly lines and transportation systems).

2. PROBLEMS WITH A "FAIRLY SMALL" NUMBER OF ALTERNATIVES

If the optimization involves selecting the best of a few alternatives, say

$$\Theta=\{\theta^{(1)}, \theta^{(2)}, ..., \theta^{(m)}\},$$

where m<30, then it may be possible to evaluate every solution and compare their performances. In the deterministic context this would be straightforward, but, since the performance must be estimated through stochastic simulations, some further analysis is needed to compare alternative solutions. Numerous different approaches have been developed to address this problem, including *subset selection*, *indifference-zone ranking and selection* (R&S), *multiple comparisons procedures* (MCPs), and *decision theoretic methods*.

Subset selection does not attempt to find the optimal solution but simply to reduce the set of candidate solutions to a (small) subset. Early work in this field have developed techniques that apply when (i) the simulation output is normal with common variance, (ii) the same number of simulation observations are used for each solution. These assumptions are rarely satisfied for simulation outputs and although many methods are quite robust with respect to the normality assumption, the assumption of common variance is quite restrictive and new techniques that do not require this assumption have been developed only recently (Nelson et al., 2001).

R&S methods aim at determining a single best (optimal) solution. To this purpose, the most common approach is to define an indifference zone δ for the performance and develop a procedure that selects a solution with performance that is within δ units of the optimal performance with a given probability. That is, if θ^* is the optimal solution and θ is the selected solution then

$$\text{Prob}\ [\ |f(\theta) - f(\theta^*)| < \delta] \geq 1 - \alpha,$$

where $1-\alpha$ is the desired probability. To achieve this guarantee, a two-stage procedure that prescribes how many simulation estimates are needed for each alternative is commonly applied. A discussion of alternative indifference-zone procedures can be found in Banks et al (2000) and Law and Kelton (2000). As subset selection and R&S procedures have complimentary functions, a natural approach is to use a subset selection procedure to remove clearly dominated solutions, followed by a R&S procedure to select the best solution (Nelson et al., 2001).

Another approach for selecting the best solution is based on MCPs which calculate simultaneous confidence intervals for

$$f(\theta^{(i)}) - f(\theta^*) \qquad (i = 1, ..., m),$$

where θ^* is as before the optimal solution (Hochberg and Tamhane, 1987). These procedures are actually closely related to the R&S procedures as indifference-zone procedures can automatically provide such confidence intervals with the width of the interval corresponding to the selected indifference zone (Matejcik and Nelson, 1995).

Most of the statistical selection procedures mentioned above involve a two stage process where in the first stage the mean and the variance of each solution are estimated and those estimates used to determine how many additional simulations are needed to make the desired selection. In implementing such methods a key issue is how much effort to put into the first stage. If it is too little an inaccurate estimate may prescribe much more simulation runs in the second stage than is really needed and vice versa too much effort in the first stage may spend more simulation time on each solution than was needed. More recently there has been considerable effort devoted to developing sequential procedures that solve this problem. Kim and Nelson (2001) and Chen et al. (2000) show that these methods perform very favorably when compared to the sequential procedures.

3. PROBLEMS WITH A "FAIRLY LARGE" NUMBER OF ALTERNATIVES

When the number of feasible solutions is "fairly large", a heuristic search need to be used in order to select a subset of good solutions to be compared through simulation. This can be done within several frameworks originally developed for solving deterministic combinatorial optimization problems.

Random search typically involves an iterative process where at each iteration the search progresses to a new (possibly better) solution, randomly chosen in the neighborhood of the current solution. The various random search methods differ because of the neighborhood structure, the way a new solution is selected and the stopping criterion.

In addition, various metaheuristics have been suggested for simulation optimization. Such methods include genetic algorithms, simulated annealing, tabu search, and neural networks (Glover and Kochenberger, 2003). Although not all these methods have a convergence guarantee, they have been quite successful when applied to simulation optimization.

Simulated annealing (SA) can be thought as a modification of the random search described above and can be adapted to simulation optimization (Haddock and Mittenhall, 1992). Starting with an initial solution, SA moves from one solution to the next, hopefully converging to the global optimum. SA attempts to avoid getting trapped into a local optima by accepting inferior solutions with certain probability. In particular, the candidate solution is always accepted if it is better than the current solution but it is also sometimes accepted even if it is inferior. The probability of accepting the inferior candidate is higher if the difference in performance is small, and it is higher if a parameter, called the temperature, is high. Usually, this temperature is allowed to decrease as the search progresses, the idea being that, after a while, no big moves up hill should be allowed and eventually no moves should be made to an inferior solution. However, in the context of simulation optimization there are indications that a constant temperature search may work as well or better (Alrefaei and Andradóttir, 1999). Ghiani et al (2004) present an iterative method that combines the robustess of SA with a statistical procedure for comparing the solutions that are generated during the search process.

Tabu search (Glover and Laguna, 1997) compares the current solution θ with its neighbors (solutions similar to θ) except those that have already visited recently (*tabu* solutions). Specifically, solutions are tabu if they require the reverse move of a recently made move, which forces the search to continue when it might otherwise get stuck at a local optimum. Although maintaining a list of tabu moves may be considered the main feature of the method, it has numerous other properties. This includes for example long term memory that allows the search to restart at a previously found good solution with a new list of tabu moves that forces a different search direction from this good starting point. More details on using tabu search for simulation optimization can be found in Glover, Kelly, and Laguna (1999) and April et al. (2001).

Genetic algorithms (GA) and other evolutionary methods work with a population of solutions rather than a single solution. Thus, at each iteration, a new population is generated on the basis of the current solutions by means of cross-over and mutation operators. The cross-over operation typically takes two solutions having good performances from the current population and combines them to make two new solutions. This resembles an evolutionary process where two individuals are allowed to reproduce to generate offspring that resemble the parents. The mutation operator, on the other hand, takes a single high performing solution and alters it slightly.

Scatter Search (Laguna, 1997) creates a set of solutions (the *reference set*) that guarantees a certain level of “quality” and of “diversity”. The iterative process consists in selecting a subset of the reference set, in combining the corresponding solutions, through a tailored strategy, in order to create new solutions, and in improving them through local optimization algorithms. The process is repeated, with the use of diversification techniques, until certain stopping criteria are met.

Another random search metaheuristic is the *Nested Partition* (NP) method of Shi and Ólafsson (2000). This method takes a global approach to simulation optimization and generates iterative partitions of the entire feasible region. That is, at the *k*-th iteration there is some subset $\sigma(k)\subseteq\Theta$ that is considered the most promising ($\sigma(0)=\Theta$), and the method attempts to narrow the search by looking at subsets $\sigma_i(k)\subseteq \sigma(k)$, $i\in\{1, 2, ..., M\}$ of this region while simultaneously looking at the surrounding region $\Theta/\sigma(k)$. Thus, it maintains a global perspective at every stage. If one of the subsets, say $\sigma_l(k)$, is found to be best this becomes the most promising region during the next iteration ($\sigma(k+1)=\sigma_l(k)$), but, if the surrounding region is found to be best, the method backtracks ($\sigma(k+1)=\sigma(k-1)$). Statistical selection methods, such as those reviewed before, can be used to determine the amount of sampling needed from each region to assure a proper selection in each iteration. Specifically, given an indifference zone δ and the probability of selecting a solution whose performance is within δ units of the optimal performance, a statistical selection procedure is used to prescribe how many solutions need to be sampled from each of the subregions and the surrounding region.

4. PARALLELIZATION STRATEGIES

When using a metaheuristic in a simulation-optimization setting, a parallel implementation is usually needed in order to make computation time acceptable. The parallelization of a metaheuristic can be accomplished in a number of ways depending on problem structure and hardware at hand. Crainic et al (1997) classify parallel TS procedures according to three criteria: *Search Control Cardinality*, *Search Control Type*, *Search Differentiation*. The first criterion indicates whether the control of the parallel search is performed by a single processor (*1-control*, 1-C), or distributed among several processors (*p-control*, p-C). The second measure denotes the way (*synchronous* or *asynchronous*) communication is performed among processors. Synchronous communication can be independent of computation status (*rigid synchronization*, RS) or not (*knowledge-based synchronization*, KS). In asynchronous communication, a processor that finds a new best solution broadcasts a message to the other processors. In the simplest case, the single solution is sent (*collegial* communication, C) while in {*knowledge-based collegial*} communication (KC) additional information are transmitted to the receivers. Finally, the third criterion accounts for the way different searches are carried on. Four alternatives are available: *Single Initial Point - Single Strategy* (SPSS) if a single search is performed; Single Initial Point - Multiple Strategies (SPMS) when each processor carries out a different search starting from the same initial solution; Multiple Initial Points - Single Strategy (MPSS) if each processor performs the same search starting from different initial solutions; Multiple Initial Points - Multiple Strategies} (MPMS) if each processor is in charge of a different search starting from a different initial solution. In SPMS and MPMS, the searches may be performed by different algorithms or, more commonly, by the same algorithm with different parameters. Not every combination of such parameters is feasible (e.g., 1-C/C/MPMS does not make any sense). The most common configurations are:

(i) 1-C/RS/SPSS (or *master-slave*) strategy. A single search is performed by a single processor (called *master*) which dispatches time-consuming tasks to the other processors (called *slaves*). There is no communication among the slaves.

(ii) 1-C/KS/SPSS strategy. Compared to 1-C/RS/SPSS, the slaves (which still do not communicate among themselves) can be required by the master to stop computing. This is usually the case when each slave is assigned a relatively simple search.

(iii) p-C/RS/MPSS, p-C/RS/SPMS and p-C/RS/MPMS strategies. Several independent searches (with different initial points or parameters) are executed in parallel. There is no communication among processors except at the end of the computations when the best solution is selected.

(iv) p-C/C/SPMS, p-C/C/MPSS and p-C/C/MPMS strategies. Each processor performs a different search. Once a processor finds a new best solution, it sends it to the other processors that re-initialize their searches.

Ghiani et al (2004) have implemented three parallelization strategies in a SA framework where solutions are compared by means of the Rinott (1978) procedure. The first implementation is a 1-C/RS/SPSS master-slave approach in which different simulation runs are executed in parallel. The remaining parallelization approaches are two multi-thread synchronous procedures in which, at each synchronization point, the new current solution is the *best* solution found so far or a *random* solution, respectively. Computational results showed that the multi-thread parallelization were able to provide slightly superior solutions (less than 3% on average) at the expense of much larger computing times (often several hundred hours even for moderately sized instances). They also showed that the *speed-up* (the ratio between the computing time provided by the sequential code and the computing time of the parallel code), which is ideally equal to the number of processors, decreases as the number of processors increases. When 2 or 4 processors were used, the speed-up was close to its ideal value; with 8 processors, the speed-up was relatively close to its ideal value; instead, with 16 processors the ideal speed-up was rarely achieved. The performance deterioration is due to two main reasons:

- if the number of processors increases then comunication overhead between master and slaves increases;
- during the first-stage of the Rinott procedure, workload is perfectly balanced among processors since the initial number of simulation runs for each is a multiple of the number of processors); however, in the second-stage this isn't always true. As a result, if the number of additional simulation replications is small, the more processors are used the more processors remain idle on average.

5. COMMERCIAL CODES

At present, nearly every commercial discrete-event simulation software package contains a module that performs some sort of "optimization" rather than just pure statistical estimation (see Table 1). This is in contrast with the status in 1990, when no package included such an option.

Table 1 - Optimization for Simulation: Commercial Software Packages

Optimization package	Simulation platform	Primary search strategy
AutoStat	AutoMod	Genetic algorithm, evolutionary strategy
Optimiz	Simul8	Neural network
OptQuest	Arena, Crystal Ball, etc.	Scatter search, tabu search, neural network

SimRunner	ProModel	Genetic algorithm, evolutionary strategy
Optimizer	Witness	Simulated annealing, tabu search

The optimization routine in *AutoStat* (Bitron, 2000) incorporates an evolutionary strategy algorithm (a genetic algorithm variation) and handles multiple objectives by requiring weights to form a fitness function. For each input variable the user wishes to optimize, the user specifies a range or set of values. For each performance measure, the user specifies its relative importance (with respect to other performance measures) and a minimization or maximization goal. The user also specifies the number of simulation replications to use for each iteration in the search algorithm. Further options include specifying the maximum number of total replications per configuration, the number of parents in each generation, and the stopping criteria, which is of two forms: termination after a maximum number of generations or when a specified number of generations results in less than a specified threshold level of percentage improvement. While the optimization is in progress, the software displays a graph of the objective function value for four measures as a function of the generation number: overall best, best in current generation, parents' average, and children's average. When complete, the top 30 configurations are displayed, along with various summary statistics from the simulation replications.

Optmiz proceeds using neural networks: "Optmiz searches for the best solution. Give Optmiz information about what to optimize (maybe a service level of 95%). Give it a list of the resources and other variables you are prepared to see change (maybe some factors are fixed but you could buy more machinery, or some types of labor). You can also give constraints on how much these factors are allowed to change. Optmiz uses Simul8's 'trials' facility multiple times to build an understanding of the simulation's 'response surface' (the effect that the variables, in combination, have on the outcome). It does this very quickly because it does not run every possible combination! It uses neural network technology to learn the shape of the response surface from a limited set of simulation runs. It then uses more runs to obtain more accurate information as it approaches potential optimal solutions."

OptQuest is a stand-alone optimization software routine that can be bundled with a number of commercial simulation environments, such as Arena and Crystal Ball. The algorithm incorporates a combination of strategies based on scatter search and tabu search, along with neural networks for screening out candidates likely to be poor. Scatter search is also a population-based evolutionary search strategy like GAs. However, Glover, Kelly, and Laguna (1999) claim that whereas naïve GAs produce offspring through random combination of components of the parents, scatter search produces offspring more intelligently by incorporating history (i.e., past evaluations). In other words, diversity is preserved, but natural selection is used in reproduction prior to being evaluated. This is clearly more important in the simulation setting, where estimation costs are so much higher than search costs. The neural network serves as a metamodel representation. Since it is clearly a rough approximation, both in approximating the objective function and in the uncertainty associated with the simulation outputs, OptQuest incorporates a notion of a *risk* metric, defined in terms of standard deviations. If the neural network predicts an objective function value for the candidate solution that is worse than the best solution up to that point by an amount exceeding the risk level, then the candidate solution is discarded without performing any simulation.

6. CONCLUSIONS

In this paper we have surveyed the main issues related to simulation optimization, with an emphasis on problems with discrete decision variables. Our analysis of the literature have shown that although several articles on this topic have been published, there is large room for improvements. In particular, we have underlined that very limited computational results have been obtained so far, so that it is not clear at present which algorithmic strategies should be used in real-world applications. In addition, we have remarked that devising tailored parallelization strategies for simulation optimization algorithms is still in its infancy although it is expected to be very beneficial in practice. Finally, we have surveyed commercial software packages which have recently started to include some (naive) simulation optimization tools.

REFERENCES

M. H. Alrefaei and S. Andradottir. 1999. A simulated annealing algorithm with constant temperature for discrete stochastic optimization. Management Science 45, 748-764.

Andradóttir, S. 1998. Simulation optimization. Chapter 9 in J. Banks, ed. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*. John Wiley & Sons, New York.

April, J., F. Glover, J. Kelly, and M. Laguna. 2001. "Simulation/Optimization using "Real-World" Applications," in Proceedings of the 2001Winter Simulation Conference, 134-138.

Banks, J., J. S. Carson, B. L. Nelson, D. M. Nicol. 2000. Discrete Event Systems Simulation, 3rd ed. Prentice Hall, Englewood Cliffs, NJ.

Bitron, J. 2000. Optimizing AutoMod Models with Auto-Stat. AutoFlash Monthly Newsletter, AutoSimulations, Inc., Bountiful, UT.

Chen, H. C., C. H. Chen, E. Yücesan. 2000. Computing efforts allocation for ordinal optimization and discrete event simulation. IEEE Transactions on Automatic Control 45 960–964.

M. C. Fu. 1994. Optimization via simulation: A review. *Annals of Operations Research* 53 199–248.

M. C. Fu. 2002. INFORMS Journal on Computing 14, 3, 192–215.

F. Glover, J. P. Kelly, M. Laguna. 1999. New advances for wedding optimization and simulation. In Proceedings of the 1999 Winter SimulationConference, eds. J.A. Joines, R.R. Barton, K. Kang, and P.A. Fishwick, 255-260. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

F. Glover, G. A. Kochenberger (Editors), 2003. Handbook of Metaheuristics, Kluwer, Amsterdam.

Glover, F. and M. Laguna. 1997. Tabu Search. Boston: Kluwer Academic.

Law, A.M. and W.D. Kelton. 2000. Simulation Modeling and Analysis, 3rd edition. New York: McGraw-Hill.

Haddock, J. and Mittenhall. 1992. “Simulation Optimization using Simulated Annealing,” Computers and Industrial Engineering, 22, 387-395.

M. Hartmann. An improvement on Paulson's sequential ranking procedure. Sequential Analysis, 7: 363-372, 1988.

M. Hartmann. An improvement on Paulson's procedure for selecting the population with the largest mean from k normal populations with a common unknown variance. Sequential Analysis, 10: 1-16, 1991.

Hochberg, Y. and A.C. Tamhane. 1987. Multiple Comparison Procedures. New York: John Wiley & Sons.

S.-H. Kim and B. L. Nelson. A fully sequential procedure for indifference-zone selection in simulation. ACM Transactions on Modeling and Computer Simulation, 11:251{273, 2001.

Matejcik, F.J. and B.L. Nelson. 1995. “Two-Stage Multiple Comparisons with the Best for Computer Simulation”, Operations Research, 43, 633-640.

Nelson, B. L., J. Swann, D. Goldsman, W. Song. 2001. Simple procedures for selectingthe best simulated system when the number of alternatives is large. Operations Research 49 950–963.

E. Paulson. 1964. A sequential procedure for selecting the population with the largest mean from k normal populations. Annals of Mathematical Statistics, 35:174-180.

Rinott Y. 1978. On two-stage selection procedures and related probability-in-equalities. Communications in Statistics, A7, 799-811.

Shi, L., S. Olafsson. 2000. Nested partitioned method for global optimization. Operations Research 48, 390–407.

***Gianpaolo Ghiani** is Associate Professor of Operations Research at the University of Lecce, Italy. His main research interests lie in the field of combinatorial optimization, particularly in vehicle routing, location and layout problems. He has published in a variety of journals, including Mathematical Programming, Operations Research, Operations Research Letters, Networks,Transportation Science, Optimization Methods and Software, Computational Optimization and Applications, Computers and Operations Research, International Transactions in Operational Research, European Journal of Operational Research, Journal of the Operational Research Society, Parallel Computing and Journal of Intelligent Manufacturing Systems. His doctoral thesis was awarded the Transportation Science Dissertation Award from INFORMS in 1998. He is an editorial board member of Computers & Operations Research.*

***Pasquale Legato** is an Assistant Professor of Operations Research at Engineering (University of Calabria), where he teaches courses on simulation for system performance evaluation. He has been visiting scholar at Duke University (NC, USA) and at “Galileo Ferraris” Institute (Torino, Italy). He has published on queuing network models for job shop and logistic systems, as well as on integer programming models. He has been involved in several national and international applied research projects and is serving as reviewer for some international journals. Current activities are in the integration of Simulation output analysis with Combinatorial Optimization algorithms for real life applications in Transportation and Logistics.*

***Roberto Musmanno** Graduated in Industrial Engineering, he received two annual research scholarships from the Italian National Research Council. Assistant Professor of Operations Research at University of Calabria from 1995 to 1998 and Associate Professor of Operations Research at University of Lecce from 1998 to 2001, Roberto Musmanno is Full Professor of Logistics at University of Calabria and chairman of the Management Engineering Degree at the same university. His major research interests are in logistics, network optimization and parallel computing. He has published more than 30 papers in a variety of journals. He is coauthor of the book: G. Ghiani, G. Laporte, R. Musmanno, "Introduction to Logistics Systems Planning and Control", Wiley, 2004. He is the scientific director of ParCoLab – Parallel Computing Laboratory – at the Department of Electronics, Informatics and Systems of University of Calabria. He is member of the Scientific Committee of the Italian Center of Excellence for High Performance Computing, funded by the Italian Ministry of University. He is member of the editorial board of the international journal: Computers & Operations Research (Pergamon). He has been involved in several international research projects. He was member of the Organizing Committee of international workshops in the field of high performance computing.*

***Francesca Vocaturo** Graduated in Management Engineering, she is a PhD student in Operations Research at the Department of Electronics, Informatics and Systems, University of Calabria, Italy. She is also a member of the research unit at the Center of Excellence on High Performance Computing (http://www.hpcc.unical.it). Her research interests are in the areas of Simulation, Output Analysis and Simulation -Optimization.*