Journal of Computational and Applied Mathematics 233 (2010) 2322–2331

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Generating scenario trees: A parallel integrated simulation–optimization approach☆

Patrizia Beraldi*, Francesco De Simone, Antonio Violi

Financial Engineering Laboratory, Department of Electronics, Informatics and Systems, University of Calabria, Via P. Bucci Cubo 41/C, 87036 Rende, CS, Italy

ARTICLE INFO

Article history:
Received 10 August 2009
Received in revised form 24 September 2009

Keywords:
Scenario tree generation
Parallel algorithms
Simulation
Moment matching

ABSTRACT

A crucial issue for addressing decision-making problems under uncertainty is the approximate representation of multivariate stochastic processes in the form of scenario tree. This paper proposes a scenario generation approach based on the idea of integrating simulation and optimization techniques. In particular, simulation is used to generate outcomes associated with the nodes of the scenario tree which, in turn, provide the input parameters for an optimization model aimed at determining the scenarios' probabilities matching some prescribed targets. The approach relies on the moment-matching technique originally proposed in [K. Høyland, S.W. Wallace, Generating scenario trees for multistage decision problems, Manag. Sci. 47 (2001) 295–307] and further refined in [K. Høyland, M. Kaut, S.W. Wallace, A heuristic for moment-matching scenario generation, Comput. Optim. Appl. 24 (2003) 169–185]. By taking advantage of the iterative nature of our approach, a parallel implementation has been designed and extensively tested on financial data. Numerical results show the efficiency of the parallel algorithm and the improvement in accuracy and effectiveness.

1. Introduction

Uncertainty is pervasive in everyday life and its representation in a form suitable for computation plays a crucial rule in decision-making models. If uncertainty is represented in terms of continuous random variables, computation is difficult to carry out since multidimensional integration is required. To overcome this drawback, approximation by a discrete set of outcomes (scenarios) is typically considered.

In the context of financial applications, which provides the motivation for our contribution, scenarios are used either directly or indirectly. In the former case, they are used, for example, to perform risk management, i.e. starting from a given portfolio some risk measures (such as VaR, CVaR, etc.) can be computed by rating future performances of different securities. Scenarios can be also used as the input parameters of stochastic programming models. In portfolio optimization, on the basis of the price scenarios in input to the optimization model, the optimal asset allocation in terms of risk and rewards is determined.

The literature on scenario generation is rich and different techniques have been proposed over the past decades. They range from *Sampling* methods (e.g. importance sampling, bootstrapping), to *Simulation* (as the classical Brownian motion and its variants belong to this class), from *Statistical* methods (such as principal component analysis technique, regression methods, moment matching) to other methods (e.g. clustering approaches, neural networks) [18]. All the methods mentioned above are compute intensive for both "space" and "time" requirements. Larger numbers of scenarios provide

☆ This research work was partially supported by EU FP6 IP Project "BEinGRID" and by MIUR, PRIN2007cod. 20073BZ5A5, 23.07.2007.
* Corresponding author.
E-mail address: beraldi@deis.unical.it (P. Beraldi).

0377-0427/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2009.10.017

better approximations of the underlying stochastic process, thus, producing more robust and reliable solutions. This aspect represents a very critical issue in a field as the financial one characterized by a high level of complexity and competition. In effect, the worldwide crisis and several recent bankruptcies have emphasized even more the necessity of advanced and sophisticated models able to provide reliable solutions for strategic planning and risk control. The time component becomes also a critical issue in the financial field especially when scenarios are used for tactical portfolio allocation (or even for on-line trading). For a volatile stock or bond, predictions based on current data typically dictate more confident trading decisions.

Parallel processing techniques provide an important tool to adequately cope with such computational intensive problem. This paper moves a step forward this direction by proposing a new approach for scenario generation implemented on a parallel computing system. The literature on parallel techniques for scenario generation is mainly centered on Monte Carlo Simulation. Among the contributions, we mention [1], where Monte Carlo simulation is employed to simulate price scenarios used for performing risk management. The method we propose belongs to the class of statistical methods and may be viewed as a variant of the moment-matching method originally proposed in [2]. The basis idea is to generate a scenario tree that best fits some target statistical properties (typically the first 4 moments and the correlation matrix). The method does not require a full knowledge of the underlying stochastic process and, thus, has a general validity. Targets may be defined either on the basis of historical data analysis or user's experience. If the first choice may be appropriate for long term strategic financial planning, the second one would be advisable for a shorter time horizon. In fact, for tactical planning, typically referring to the construction of an asset allocation mix following a given benchmark, the end-user may wish to express views on the future deviating from the past. Under this respect, moment matching represents a very flexible and efficient approach.

The method relies on the solution of an optimization model where both the outcomes and probabilities associated with each node of the tree are decision variables. This choice leads to a non-linear optimization problem of difficult solution even for relative small instances. To overcome this drawback two main approaches have been proposed. The first one relies on the sequential solution of smaller optimization problems associated with each node of the scenario tree. However, such an approach lacks a direct control of the statistical properties defined over all the scenario tree. The second one is based on the definition of a heuristic procedure that approximately solves the original problem. By means of different transformations, starting from univariate distributions the joint ones are obtained by imposing the target correlation matching. For this approach there is no guarantee on the approximation error. Furthermore, all the scenarios have the same probability of occurrence. Our approach relies on the idea of solving a simplified version of the optimization model where the decision variables are represented by the scenario probabilities. Instead, the outcomes associated with each node are generated by a simulation procedure. This hybrid approach is similar to the one proposed in [3]. The main differences are related to the technique used for scenario generation and to the formulation of the optimization model. Even though in principle any scenario generation technique can serve the purpose of generating the outcomes associated with each node, we have chosen to implement the heuristic proposed in [4] since it tries to match for construction the defined statistical properties. In addition, our optimization model works with the probability associated with scenarios with a consequent reduction of the size of an optimization model defined on the basis of node probabilities. We also observe that specific constraints bounding the probabilities have been considered in our model in order to avoid the elimination of scenarios (those having probability 0) or the collapse of the entire tree on a singleton scenario (with probability 1).

The rest of the paper is organized as follows. Section 2 describes the proposed approach for scenario tree generation. The method is well suitable for parallel implementation and Section 3 provides a description of the parallel approach. Section 4 is devoted to the computational experiments. First the test case is described and then numerical results are presented and discussed by showing the performance of the parallel algorithms and the improvement in terms of solution quality. Conclusions are reported in Section 5.

2. The integrated approach

Let $\xi = \{\xi_t\}_{t=1}^{\mathrm{T}}$ be a multivariate stochastic process defined on the probability space (Ω, F, P). In the case of discrete distributions (real or approximated) ξ is typically represented by a scenario tree, where each node n at level t corresponds to possible outcome of ξ_t. We denote by $a(n)$ the unique predecessor of node n and we assume that it may have a certain number of successors (children). Nodes at the horizon T are referred to as leaf nodes and are used to identify the scenarios. More specifically, a scenario is a path from the root note (generally denoted by 0) to a leaf node and, thus, it represents a joint realization of the uncertain problem parameters over all time steps $t = 0, 1, \ldots, T$. We shall denote by S the number of scenarios. With each scenario s is associated a probability of occurrence p_s satisfying the fundamental probability axioms [17]. We note that the values of p_s can be also expressed by the conditional probabilities associated with intermediate nodes of the tree.

The Fig. 1 shows a binomial scenario tree for a time horizon of length 3. We observe that the tree topology (i.e. number of nodes per stage) can vary according to problem specific requirements and could not be constant through the tree. One of the strategies often adopted in strategic financial planning is to use an extensive branching at the beginning of time horizon and a relatively poor branching at the last levels of the tree.

Besides the topology, the end-user may specify some statistical properties the scenario tree should have. As in [2] we shall consider the first four moments and the correlation matrix. Such values can be either determined on the basis of the historical data analysis or may reflect decision maker personal views on the future evolution of the random quantities.

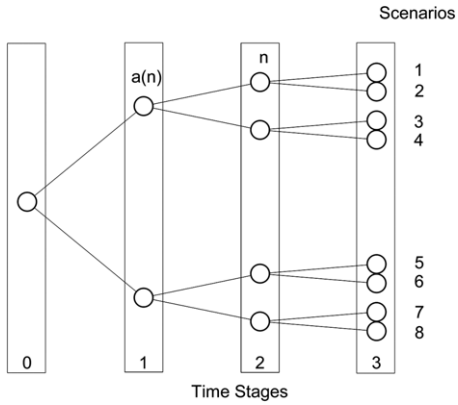

Fig. 1. Scenario tree.

Our method can be regarded as an integrated approach where the simulation phase provides the input parameters for an optimization model aimed at defining the scenarios probability matching the prescribed targets. The two main steps, simulation and optimization, are repeated for a certain number of iterations.

The approach can be summarized according to the following scheme.

Step 0 (Initialization). Read input parameters referring to the tree topology and target statistical properties. Set the maximum number of iterations $K_{\max}$, the iteration counter $k_{\text{iter}} = 0$ and the incumbent value for optimal distance from targets $\bar{D} = \infty$.
Step 1 (Termination). If $k_{\text{iter}} = K_{\max}$, then Stop.
Step 2 (Simulation). Generate the outcomes for each node of the tree.
Step 3 (Optimization). Solve an optimization problem and determine D.
Step 4 (Test). Verify if $D < \bar{D}$. In such a case update the incumbent value $\bar{D} = D$. Set $k_{\text{iter}} = k_{\text{iter}} + 1$ and go to Step 1.

In what follows we specify how the main steps, simulation and optimization, are performed.

2.1. Simulation

The generation of the random vector outcomes by a simulation approach represents the first key ingredient of the proposed scenario tree generation method. Even if in principle any simulation technique could serve this purpose, the use of methods guaranteeing good statistical properties would be preferable.

Our method implements the approach presented in [4] and inspired in [5–7]. The basic idea is to work with the marginal distributions gaining the joint one by means of different transformations consistent with the correlation matrix. In particular, the algorithm works as follows: for each component of the random process generate an univariate random variable from a normal standard distribution. Compute the first 12 moments of each variable and apply a "cubic" transformation to create variables with first moments close to target. Apply a "matrix" transformation to create multivariate random variables with the specified correlation matrix. Algorithmic details can be found in [4]. Generally simulation is a highly valuable tool for tackling practical problems, but it not sufficient by itself to yield the quality of outcomes desired. A step forward is represented by an approach that joins simulation and optimization. Presentations of simulation–optimization techniques may be found in the survey papers in [8,9].

In our case, the procedure just described is only approximate and there is no guarantee about the error level. The optimization problem described in Section 2.2 aims at reducing the error manipulating probabilities of each scenario.

2.2. Optimization

Let us denote by I the size of the stochastic process and let x_{it}^s identify the simulated value of the random component i at stage t under scenario s. On the basis of these values, we may determine the statistical properties that will enter in our model. In particular, we shall consider the first four moments indexed by l ($l = 1, \ldots, 4$) and the correlation defined as:

$$\bar{M}_i^l = \sum_{s=1}^{S} p_s M_{is}^l \quad i = 1, \ldots, I, \ l = 1, 2, 3, 4 \tag{1}$$

$$\bar{C}_{ij} = \sum_{s=1}^{S} p_s C_{ijs} \quad i, j = 1, \ldots, I \tag{2}$$

where

$$M_{is}^1 = \mu_{is} = \frac{1}{T}\sum_{t=1}^{T} x_{it}^s \quad s = 1, \ldots, S,\ i = 1, \ldots I \tag{3}$$

$$M_{is}^2 = \sigma_{is}^2 = \frac{1}{T}\sum_{t=1}^{T} (x_{it}^s - \mu_{is})^2 \quad s = 1, \ldots, S,\ i = 1, \ldots, I \tag{4}$$

$$M_{is}^3 = SK_{is} = \frac{1}{T\sigma_{is}^3}\sum_{t=1}^{T} (x_{it}^s - \mu_{is}^2)^3 \quad s = 1, \ldots, S,\ i = 1, \ldots, I \tag{5}$$

$$M_{is}^4 = K_{is} = \frac{1}{T\sigma_{is}^4}\sum_{t=1}^{T} (x_{it}^s - \mu_{is})^4 \quad s = 1, \ldots, S,\ i = 1, \ldots, I \tag{6}$$

$$C_{ijs} = \sigma_{ijs} = \frac{1}{T}\sum_{t=1}^{T} \frac{(x_{it}^s - \mu_{is}) * (x_{jt}^s - \mu_{js})}{\sigma_{is} * \sigma_{js}} \quad s = 1, \ldots, S,\ i, j = 1, \ldots, I. \tag{7}$$

With these definitions the mathematical model can be formulated as follows:

$$\max \sum_{l=1}^{4} \omega_l \sum_{i=1}^{I} (\bar{M}_i^l - \tilde{M}_i^l)^2 + \omega_5 \sum_{i=1}^{I}\sum_{j=1}^{I} (\bar{C}_{ij} - \tilde{C}_{ij})^2 \tag{8}$$

s.t.

$$\sum_{s=1}^{S} p_s = 1 \tag{9}$$

$$L \le p_s \le U \quad s = 1, \ldots, S \tag{10}$$

where $\tilde{M}_i^l$ is the target value for moment l for component i and $\tilde{C}_{ij}$ is the target value for correlation between i and j.

The objective function (8) is defined as the weighted (by w_l) sum of squared mean errors of multivariate distribution moments with the respect to the target values. Condition (9) is required by the nature of decision variables p_s, while the constraint family (10) imposes lower and upper bounds on the probabilities values. These latter constraints, not explicitly included in other moment-matching models, avoid that the probability associated with some scenarios may take the value 0 (i.e. the corresponding scenario is removed) or 1 (i.e. the entire scenario tree collapses in an unique path).

The previous model belongs to the class of non-linear programming problems. Despite model presented in [2], the decision variables are associated with scenarios rather than nodes with a consequent reduction of the problem size. Nevertheless, nodes probabilities can be determined by the scenario ones by applying the precedence relation and simply recalling the relation between conditional and absolute probabilities.

3. The parallel implementation

The scenario tree generation method presented in Section 2 relies on the solution of several instances of the same optimization problem defined starting from different input data determined thought the simulation step. As discussed in the original contribution [4], even though the convergence of the heuristic approach cannot be demonstrated theoretically, there is an empirical evidence that after a certain number of iterations the solution becomes stable.

The nature of the method suggests a straightforward parallelization strategy that, despite its simplicity, can lead to an attractive reduction in the computational time and improvement of the accuracy level. The analysis of the procedure shows that the main computational effort is required in Step 2 and Step 3 where the generation of the outcomes and the solution of the corresponding optimization model (8)–(10) are performed.

The method is suitable for running on a parallel computing environment since the computational workload can be efficiently split among available computing units. The parallel implementation we propose has been designed by considering a Master/Slave paradigm. The following software code reports a sketch of the parallel procedure.

The master process reads input data (i.e. tree's topology, number of random variables and targets), splits the computational workload and distributes it among slaves. The master also participates in the execution of a given fraction of workload. In order to guarantee a balanced distribution of the computational workload among the available computing units (N including master) the following strategy has been implemented. Each computing node executes $V = \lfloor K_{\text{Max}}/N \rfloor$ iterations (rounded up to the nearest integer). Furthermore, an additional iteration is assigned if the difference $K_{\text{Max}} - V$ is greater than 0. Once completed its workload, each computing node sends to all the other nodes only partial information related to the achieved distance from targets. On the basis of received values and the computed ones, each computing node is able to establish if it has processed the "best" scenario tree, in terms of minimum distance from targets. If this is the case, it saves the whole scenario tree. In the case of multiple "best" trees, the processor with minimum order number is deputed to store the solution. This strategy allows to reduce the data transmission among processors and preserves from high idle times. We

note that in the described parallelization strategy the synchronization overhead is limited, thus good performances can be obtained also on low-budget parallel machines without shared memory.

```
/* header files */
#include <mpi.h>
#include "MM_Utility.h"
#include "newmat10/newmat.h"
#include "newmat10/newmatap.h"
#include "newmat10/newmatio.h"
...
#define MASTER 0
...
int main(int argc, char* argv[]){

        /* variable declaration */
        int sons;
        int levels;
        int random_variables;
        int iterations, my_iterations;
        int numproc, myid, best_id;
        double my_best_distance,my_distance,best_distance;
        Matrix Scenario_Tree;
        Matrix Probability;
        Matrix My_Best_Scenario_Tree;
        Matrix My_Best_Probability;
        int len_specifications=compute_len(random_variables);
        double *vector_specifications = new double[len_specifications];
        ...
        /* MPI init called (first) by all processes */
        MPI_Init(&argc, &argv);
        MPI_Comm_size(MPI_COMM_WORLD, &numproc);
        MPI_Comm_rank(MPI_COMM_WORLD, &myid);

        /* This next statement is only carried out by the MASTER process.
        This process reads specifications in input and put its in a vector */
        if(myid=MASTER){

                read_specifications(vector_specifications, sons,
                                        levels, random_variables, iterations);
        }

        /* MPI_Bcast is called by all processes. Master sends to other processes
        a vector with the specifics of problems */
        MPI_Bcast(vector_specifications, len_specifications,
                                MPI_DOUBLE, MASTER , MPI_COMM_WORLD);

        /* Each process compute iterations to perform */
        my_iterations=load_balancing(myid, iterations);

        /* Each process set to one million*/
        my_best_distance=1000000;

        /* Each process starts to perform scenario generation procedures */
        for(int iter=0; iter < my_iterations; iter++){

                /* Simulation phase */
                Scenario_Tree = Simulation(vector_specifications);

                /* Optimization phase */
                Probability   = Optimization(vector_specifications,Scenario_Tree);

                /* Compute distance */
                my_distance =   compute_distance(Scenario_Tree, Probability);

                /* Compare this distance with the best one */
                if(my_distance < my_best_distance){
                        my_best_distance          = my_distance;
                        My_Best_Scenario_Tree = Scenario_Tree;
                        My_Best_Probability   = Probability;
                }
        }

        double *All_distances = new double[numproc];
```

```
        /* MPI_Allgather is called by all processes. Each sent its result
        to others. */

        MPI_Allgather(&my_best_distance, 1, MPI_double, All_distances,
                                            numproc , MPI_DOUBLE, MPI_COMM_WORLD);

        /* Compute Best Distance */
        best_id = compute_best_id(All_distances);

        /* The process that has best result write in output its result */
        if(myid=best_id){

                write_result(my_best_distance, My_Best_Scenario_Tree,
                                                    My_Best_Probability);
        }

    MPI_Finalize();
    return 0;
}
```

4. Computational experiments

In this section we report on the computational experiments carried out to evaluate the performance of the proposed parallel scenario generation approach. The algorithm has been implemented in C/C++ and uses several scientific libraries to perform the operations required both in the simulation and optimization phase. In particular, the GNU Scientific Library [10] has been used to solve the system of non-linear equations which is at the core of cubic transformation. The NEWMAT [11] library has been used to perform the Cholesky decomposition in the matrix transformation. Finally, the NAG [12] library has been employed to solve the non-linear problems in the optimization phase.

The parallel code has been implemented by using the MPI paradigm [13,14]. In particular, we have adopted the MPICH2 [15] implementation which provides a library of C and C++ functions for easily developing parallel programs. All test have been carried out on a NEC TX7 with 32 CPUs Itanium II @1000 MHz and 64 GB memory, running Suse Linux Enterprise Server.

In what follows, we first introduce the testing environment and then we present and analyze numerical results.

4.1. Test case

We have considered the generation of scenario trees for the returns of financial indices listed on different markets. Historical data are represented by monthly returns for the time horizon from January 1999 to October 2006. We have considered different tree topologies, all with the same time horizon of five months with monthly steps. Different tests have been generated by varying the number of branches for node and the size of the random process.

In particular, we have considered symmetric trees with a number of branches for node equal to 3, 4, 5, 6 and 7, thus resulting in 243, 1024, 3125, 7776, 16 807 scenarios, respectively. Target moments have been obtained from historical data analysis, even if in general end-user may wish to specify these values according to his/her own perspective. Weights associated with each target in the objective function are specified in Table 1. Other choices denoting different moment priority can be effective as well, according to user requirements.

In the optimization phase, in order to avoid degenerative situations, both lower and upper bounds on the scenario probabilities have to be fixed. In particular, starting from a value $r = 1/S$, upper and lower bounds have been obtained as $U = r \times \sqrt{S}$ and $L = r \times 1/\sqrt{S}$, respectively.

The user can also specify the maximum number of iterations to perform, according with tree dimension and desired accuracy level. As we will show later in this section, it is empirically evident that after a certain number of iterations solutions seem to converge.

4.2. Numerical results

A first set of experiments has been carried out to evaluate the performance of the sequential version of the scenario generation approach. Table 2 shows the average computational time per iteration (defined as the sum of the time required by both Step 2 and Step 3) for the different test cases defined as function of the size of the random process and scenarios number.

The analysis of the results shows that the computational time mostly depends on the number of scenarios and quite poorly on the size of the random vector. This is mainly due to the higher complexity of the optimization phase with respect to the simulation one. In fact, in the optimization problem the number of the decision variables is related to the number of scenarios. Nevertheless, higher number of scenarios guarantees, as shown later in the section, a better quality of the solutions. It should be emphasized that the whole computing time also depends on the number of iterations performed. The

Table 1
Weights' values.

Moment	Weight value
Mean	0.22
Std. Dev.	0.22
Kurtosis	0.22
Skewness	0.12
Correlation	0.22

Table 2
Average solution time per iteration (s).

Size	Scenarios				
	243	1024	3125	7776	16 807
5	0.23	3.24	19.27	206.48	12 127.21
10	0.28	3.36	20.20	231.71	12 544.84
20	0.30	3.47	20.94	245.27	12 901.43

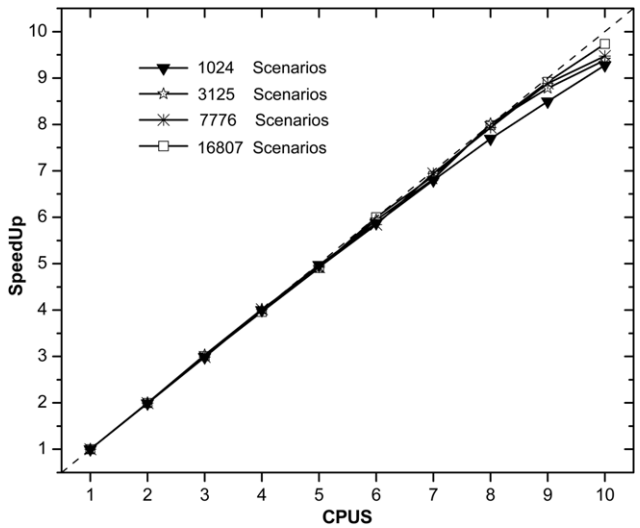

Fig. 2. Speed-up of the scenario generation approach.

choice of this latter value is up to the end-user and it is typically performed by considering the trade-off between quality and computing burden. However, the high computing time clearly underlines the necessity to take advantage of parallel systems to generate accurate solutions. Fig. 2 shows the speed-up values obtained for the test cases with random size 10, as function of the number of processors.

As evident, speed-up values are very close to the linear one for all the test cases. As regards the efficiency, the values obtained are very close to 1 as shown in Fig. 3. Similar results have been obtained for all the other test problems.

Another set of experiments has been carried out to empirically test the convergence of the proposed procedure. In particular, we have considered three test problems defined by a random vector size of 10 and a number of scenarios of 1024, 7776 and 16 807, respectively. For each test problem, we have run the whole procedure several times. In order to normalize the distance from targets, we have used an error measured as:

$$\epsilon \equiv \sqrt{\frac{\bar{D}}{\sum_i \sum_l (\tilde{M}_i^l)^2}}.$$

Fig. 4 shows the results.

As we can see, for each test case the error decays very fast and becomes stationary in few hundred iterations. Moreover, the figure confirms that larger scenario trees can better match statistical properties. We also note that the proposed procedure is robust in that for all the considered test problems the error is less than 6.5%.

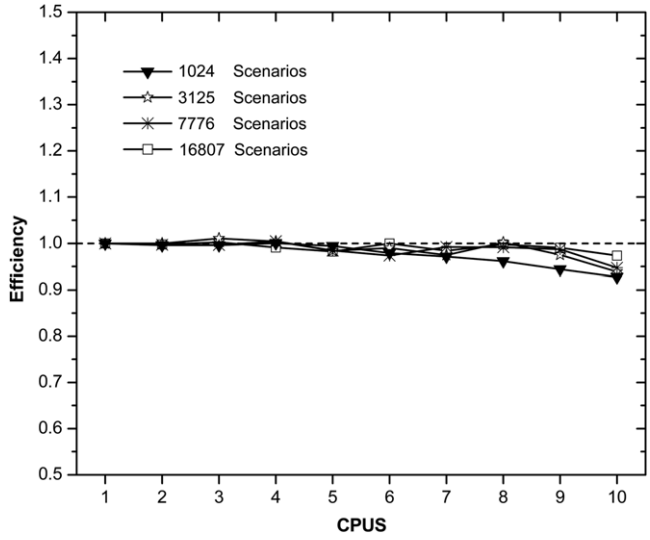

Fig. 3. Efficiency of the scenario generation approach.

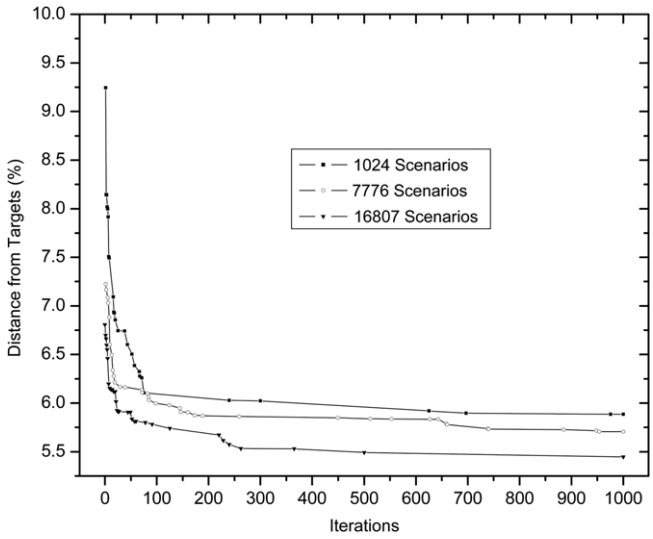

Fig. 4. Convergence of solutions for different numbers of iterations.

Other experiments have been carried out to validate the effectiveness of the proposed scenario generation procedure. To this aim we have performed a backtesting analysis by comparing the simulated evolution of a portfolio equally distributed among 5 indices with the evolution obtained by considering the real data. The simulated scenarios refer to the time horizon from October 2005 to March 2006, for which real data are available. Fig. 5 shows simulated evolutions and the real evolution for a different number of scenarios. It is worthwhile noting that generated trees represent well the real evolution and, as excepted, the higher the scenario number the better the effectiveness.

As mentioned above, in the financial field scenarios are generally used to perform risk management. One of the most widely used measures is represented by the Conditional Value at Risk (CVaR), which is known to be a more consistent risk measure than Value at Risk (VaR) (see, for example, [16]). For a given confidence level α, the main difference between CVaR and VaR consists in that VaR concerns about the maximum loss that can be observed with probability α, whereas CVaR concerns about the expectations of losses exceeding VaR. Therefore, CVaR can provide more extensive benefits information than VaR.

We have evaluated the CVaR at different confidence level α for a portfolio with of 5 indices with an increasing number of scenarios. In particular, for each test we have computed the average and the standard deviation of the CVaR values on 25

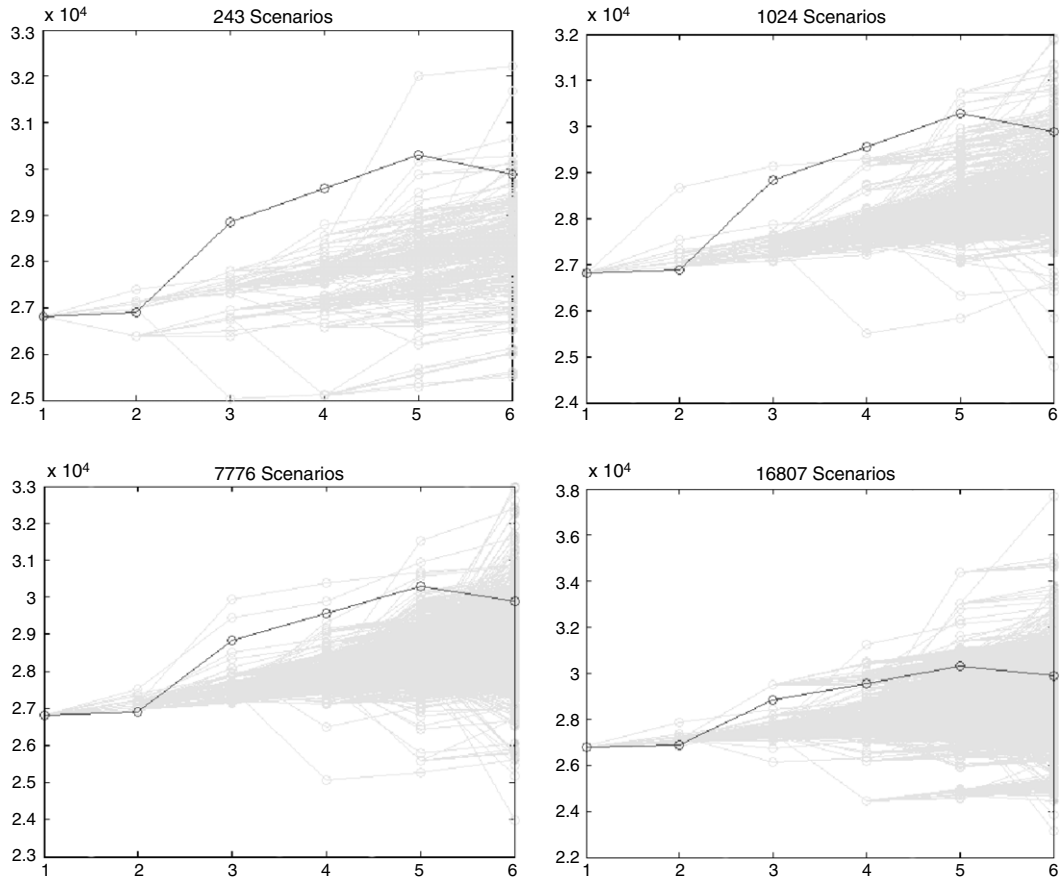

Fig. 5. Scenario tree vs. real evolution.

Table 3
Average CVaR values (%).

Scenarios	$\alpha = 99\%$	$\alpha = 95\%$
243	1.89	1.32
1 024	1.76	1.24
3 125	2.23	1.96
7 776	2.10	1.72
16 807	2.57	1.83

different scenario trees. Table 3 reports the average CVaR values for two different confidence levels, whereas the relative error is depicted in Fig. 6.

As evident, the error decreases as the scenario number is increased. This confirms that the proposed approach provides accurate solutions and represents a helpful tool for risk evaluation and control.

5. Conclusions

In this paper we propose a scenario generation approach based on the idea of integrating simulation and optimization techniques. In particular, simulation is used to generate outcomes associated with the nodes of the scenario tree which, in turn, provide the input parameters for an optimization model aimed at determining the scenarios' probabilities matching some predefined targets. A parallel implementation of the method is provided and extensively tested on financial data. The numerical results have shown the efficiency of the parallel implementation and the high quality of generated scenarios in terms of accuracy and effectiveness.

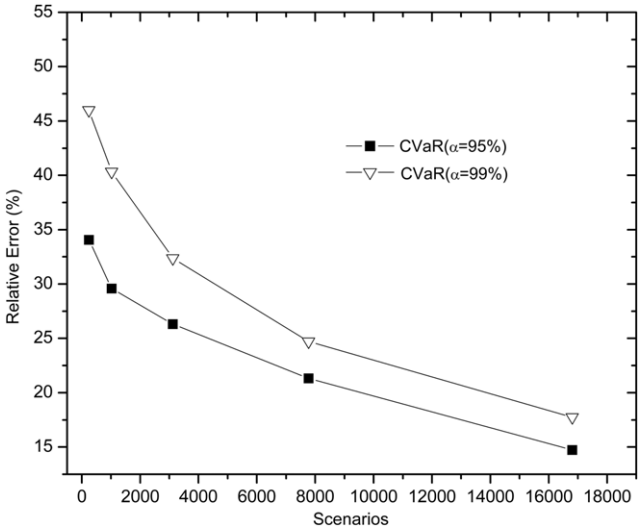

Fig. 6. Relative error in CVaR evaluation.

References

[1] R. Moreno-Vozmediano, K. Nadiminti, S. Venugopal, A.B. Alonso-Conde, H. Gibbins, R. Buyya, Portfolio and investment risk analysis on global grids, J. Comput. System Sci. 73 (2007) 1164–1175.
[2] K. Høyland, S.W. Wallace, Generating scenario trees for multistage decision problems, Manage. Sci. 47 (2001) 295–307.
[3] N. Gülpinar, B. Rustem, R. Settergren, Simulation and optimization approaches to scenario tree generation, J. Econom. Dynam. Control 28 (2004) 1291–1315.
[4] K. Høyland, M. Kaut, S.W. Wallace, A heuristic for moment-matching scenario generation, Comput. Optim. Appl. 24 (2003) 169–185.
[5] A.I. Fleishman, A method for simulating nonnormal distributions, Psychometrika 43 (1978) 521–532.
[6] P.M. Lurie, M.S. Goldberg, An approximate method for sampling correlated random variables from partially-specified distributions, Manage. Sci. 44 (1998) 203–218.
[7] C.D. Vale, V.A. Maurelli, Simulating multivariate nonnormal distributions, Psychometrika 48 (1983) 465–471.
[8] S. Andradttir, Simulation optimization, in: J. Banks (Ed.), Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice, John Wiley & Sons, New York, 1998 (Chapter 9).
[9] M.C. Fu, Optimization for simulation: Theory vs. practice, INFORMS J. Comput. 14 (3) (2002) 192–215.
[10] http://www.gnu.org/software/gsl/.
[11] http://www.robertnz.net/nm10.htm.
[12] http://www.nag.co.uk.
[13] W. Gropp, E. Lusk, A. Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface, 2nd ed., The MIT Press, Cambridge, 1999.
[14] M.J. Quinn, Parallel Programming in C with MPI and OpenMP, McGraw-Hill, New York, 2003.
[15] http://www.mcs.anl.gov/research/projects/mpich2staging/balaji/mpich2/index.php.
[16] P. Artzner, F. Delbaen, J.M. Eber, D. Heath, Conherent measures of risk, Math. Finance 9 (1999) 203–228.
[17] J. Dupacová, G. Consigli, S.W. Wallace, Scenarios for multistage stochastic programs, Ann. Oper. Res. 100 (2000) 25–53.
[18] S. Mitra, Scenario Generation for Stochastic Programming, White Paper, Optirisk Syst., UK, 2006.