Transactions on Information and Communications Technologies vol 3, © 1993 WIT Press, www.witpress.com, ISSN 1743-3517

Parallel implementation of simulated annealing by distributed memory systems

A. Consiglio[a], A. Genco[b], A. Pecorella[c], G. Pecorella[d]
[a]*Istituto di Matematica per la Ricerca Operativa*
[b]*Dipartimento di Tecnologia e Produzione Meccanica*
[c]*Centro Universitario di Calcolo*
[d]*Istituto di Automatica e Sistemistica*
Università di Palermo - Italy

ABSTRACT

The aim of this paper is to provide a parallel strategy to be used when implementing the Simulated Annealing process by distributed memory systems.
It focuses the attention on the portfolio selection problem with dynamic and integer constraints on the variables. However, this method can be extended to any dynamic optimisation problem as the optimal control.
In the first part it develops the sequential algorithm of Simulated Annealing stressing some features inherent in the application itself: the structure of the neighbourhood of the solution, the starting value and the decreasing schedule of the control parameter.
In the second part it deals with the formulation of a parallel strategy for the algorithm. It starts from the consideration that an optimisation technique with a probabilistic searching criterion cannot give a reliable result in a single execution. Therefore it proposes a strategy performing a number of independent concurrent Simulated Annealing processes applied to different initial configurations of the same problem.
Finally, it proposes a co-operation policy among processes, by means of a reduced number of communications. This policy enhance the efficiency of the parallel computing system.

INTRODUCTION

Investment diversification is known as the best way to reduce the intrinsic risk of individual financial assets.

In early 1959 [1], Markowitz provided the theoretical framework for this topic, introducing the variance of a portfolio as measure of the total risk and, thus resolving the quadratic programming problem obtained.
Many authors have improved the Markowitz's original work by adding or modifying some assumptions that bore no direct relation to the real mechanisms of stock markets.
Two of these assumptions are: one, the hypothesis of perfect divisibility of the shares and the other the one-period structure of the model that limits the applicability of Markowitz's approach.
The case study of this paper consists in offering a solution for a multi-period model of portfolio selection in which the variables are constrained to take integer values. This problem is a very complex one; only a heuristic technique can give useful results in terms of computational time and memory requirements.
Simulated annealing (SA) is a well-known algorithm used to solve combinatorial and discrete optimisation problems. We implemented a modified version of SA to portfolio selection and, furthermore, we developed a parallel version of this method in order to distribute the computation onto a network of communicating processors.

MULTI-PERIOD PORTFOLIO SELECTION

A multi-period model of portfolio selection was proposed by Sengupta [2]. In this model the portfolio is represented by a discrete-time system evolving over a specified time-horizon. The objective is to minimise a functional whose meaning is analogous to the one-period objective function. The original model has been modified in order to take into account integer constraints on variables, so we have:

$$\text{(2.1)} \qquad \text{P}: \begin{cases} \min \quad J = \sum_{k=1}^{N-1} (1+\delta)^{-k}\, \bar{x}_k' R_k \bar{x}_k \\ \quad s.t. \\ \quad \bar{x}_{k+1} = M_k \bar{x}_k + M_k L \bar{u}_k \\ \quad \frac{1}{S_k} \bar{r}_k' (\bar{x}_k + L \bar{u}_k) \geq \mu_k \\ \quad \bar{p}_k' L \bar{u}_k \leq C_k \end{cases}$$

where:

$\bar{x}_k$ is a *n-dimensional* vector, whose components (x_{ik}) represent the amount of capital allocated on stocks at time *k*;

$\bar{u}_k$ is a *n-dimensional* vector, whose components (u_{ik}) represent the number of lots purchased at time *k*;

$\bar{r}_k$ is a *n-dimensional* vector, whose components (r_{ik}) are the expected return on stocks at time *k;*

L is a *[n x n]* diagonal matrix containing the minimal amount of stocks that can be exchanged;

R_k is the variance-covariance matrix of the returns of the stocks included in the portfolio;

M_k is a *[n x n]* diagonal matrix containing the elements $m_{ik} = (1 + r_{ik})$ which correspond to the expected earn for a unit of capital invested;

$\bar{p}_k$ is a *n-dimensional* vector, whose components p_{ik} are the expected prices at time k

δ, S_k, C_k and μ_k are, respectively, the rate of interest, the amount of capital invested at time *k*, the amount of capital available at time *k*, the total return of the portfolio at time *k*.

The objective is to minimise a future stream of risks discounted at the interest rate δ. The first constraint describes the dynamics of the portfolio. The amount of capital at time *k+1* is made up of the capital allocated in the portfolio at time *k* plus the interest ($M_k\ \bar{x}_k$) and by the amount of stock purchased at time *k*. The others two are a dynamic version of the return and balance constraint, assuring that the optimal investment policies yield a minimal total return for each period and that the capital invested does not exceed the capital available.

SIMULATED ANNEALING: AN OVERVIEW.

Simulated annealing was originally developed by Kirkpatrick et al. [3] to solve combinatorial or discrete optimisation problems. This method is based on the statistical mechanical annealing of solids. Annealing is the process by which a solid at high temperature is brought to a low temperature state by the gradual reduction of temperature. The algorithm describing this process was provided by Metropolis et al. [4] in early 1953. In Metropolis's algorithm a collection of configurations are randomly generated at each temperature *T,* observing that the energies of the different configurations in this set can be represented by a Boltzmann distribution given by:

$$P(\Delta E) = \exp\left(-\frac{\Delta E}{k_B T}\right) \tag{3.1}$$

where $\Delta E = E_j - E_i$.

Once a configuration is generated, as result of a small perturbation of the last one, the correspondent energy level is valued. This new configuration is accepted if the level of energy decreases; otherwise, the increment ΔE will be subjected to the probability function (3.1).

Temperature has no direct analogue in optimisation; it merely serves as a control parameter. At high temperatures, as seen from expression (3.1), the SA algorithm can traverse almost the entire space of solutions because large increases of energy solutions can be easily accepted. However, by successively lowering the temperature, the algorithm is confined to a smaller and smaller region of the space of solutions. Thus, in optimisation , SA works as follows: at high temperatures, the SA algorithm behaves more or less like a random search. The continuos accepting increases of objective the function allows the algorithm to escape from local minima. At low temperatures, only solutions that decrease the objective function will be accepted, stopping the algorithm from leaving the region that most probably contains the optimal solution.
It is important to underline that finding the optimal solution depends on different parameters in the annealing process. Unfortunately, it is very difficult to offer a standard rule about the determination of these values. They depend on the optimisation problem one is dealing with, and, for the same problem, according to the structure, we will need different values for these parameters.
Many authors [5],[6],[7] suggest a tuning stage in order to determine the parameters that make the optimisation process converge better towards the optimal solution.

APPLYING SA TO PORTFOLIO SELECTION.

The application of SA to portfolio selection is straightforward. Every investment policy can be considered as a configuration of the annealing process. In our problem the objective is to minimise the total risk of the portfolio over a time horizon. Thus, the discounted stream of the future risk will assume the same meaning as energy in the annealing process. As we have said, the algorithm evolves generating new configurations by a small perturbation of the last accepted one. To do that, we have to define the neighbourhood of the solutions and carry out the random selection inside this neighbourhood. Of course, the new investment policy must be feasible. Given a feasible investment policy x_i a new investment policy x_j is obtained by adding 1,0, or -1 to the former. If this configuration does not satisfy the constraints, a high value of risk will be attributed to it, in this way the correspondent configuration will definitely be refused and the algorithm will converge toward a feasible optimal solution.
As mentioned in the previous section, the parameters used in the annealing process affect the success of the algorithm in its search for the global optimal solution. So, it is important to select an appropriate starting value and cooling schedule for the control parameter.
The starting value of the control parameter is normally set to a value that allows about 80% of transitions to be accepted in order to make the algorithm jump from one solution to another as above mentioned. As regards the cooling schedule, if the control parameter is lowered quickly, the algorithm tends to converge towards local minima. On the other hand, too slow a reduction will produce a waste of computer time. From our experiments we have found that an

exponential rate of reduction gives a good annealing process; furthermore, an exponential decrease of the control parameter highlight a condition useful for the parallelism of the algorithm.
Taking into account the argument presented, we can provide a C-like code for SA applied to portfolio selection:

```
main()
{
   seed (); /* initialise a random generator */
   init_invest(); /* Generate a random investment policy */
   fo_old=Risk(); /* Evaluate the risk of the first investment policy */
   c_N = init_temp ; /* Assign the initial value of the control parameter */
   do{
      do{
       perturbation(); /* Perturbs the last configuration accepted*/
       fo_new=Risk();
       ΔRisk = fo_new-fo_old;
       if(ΔRisk > 0){
              Boltz = e^(-ΔRisk/c_k) ;
              if (Boltz > random [0, 1]) accept current investment policy;
              else discard it;
       }
       else accept current investment policy;
       iter ++
       } while( iter < iter_max );
       c_{k+1} = f(c_k);
       }while ( c_k > final_temp)
} /* end main */
```

PARALLEL IMPLEMENTATION

The implementation of SA on a multiprocessor system is not straightforward because of the sequential nature of the method. SA is commonly described as a sequence of Markov chains where each computation step can start only after the previous one has beencompleted.
The algorithms proposed in literature can be grouped in two classes. In the first class, one finds algorithms that try to reduce the amount of computation time in performing a single chain [5],[7],[8],[9]. In the second class, one finds algorithms basing on the idea that if an optimal solution is desired, one can run several annealing processes independently on different processors.
We focused our attention on this last class of algorithms because these strategies are the most effective ones and, furthermore, they can be improved thus obtaining super-linear speedups [10],[11],[12].

We focused our attention on this last class of algorithms because these strategies are the most effective ones and, furthermore, they can be improved thus obtaining super-linear speedups [10],[11],[12].
An important feature to investigate is the influence of the decreasing schedule of the control parameter on the parallel independent annealings.
We observed in our experiments that using a linear decreasing schedule of the control parameter, the trajectories cross each other and it is impossible to recognise the most promising annealing processes (Fig. 1).
Conversely, using an exponential decreasing rule, the annealing processes producing the best solutions maintain this behaviour up to the end and very rarely do the worst ones obtain the best solution.
Furthermore, the exponential decrease provokes an initial narrow beam of the trajectories that gradually enlarges during the middle period.

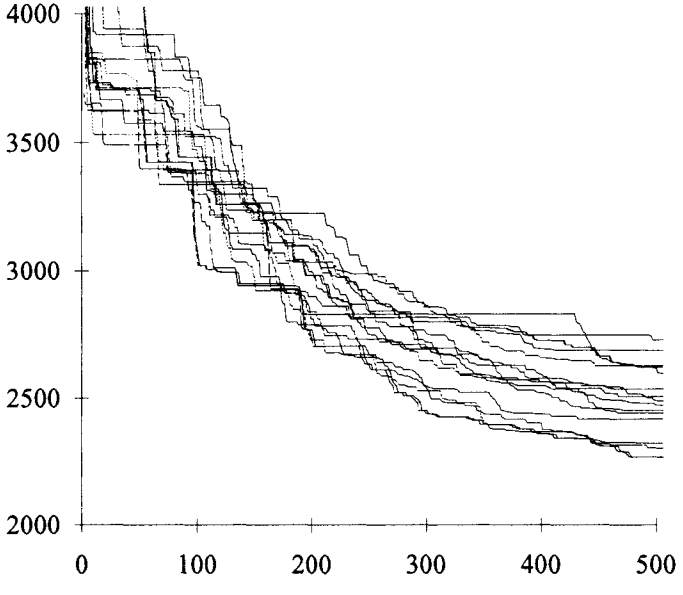

Fig. 1: linear decreasing schedule.

It can be observed that the exponential decrease produced trajectories at a lower level then the lowest one reached by the linear decrease.
If we correctly determine a suitable degree of parallelism, there is a high probability that this large beam of trajectories includes the optimal solution.

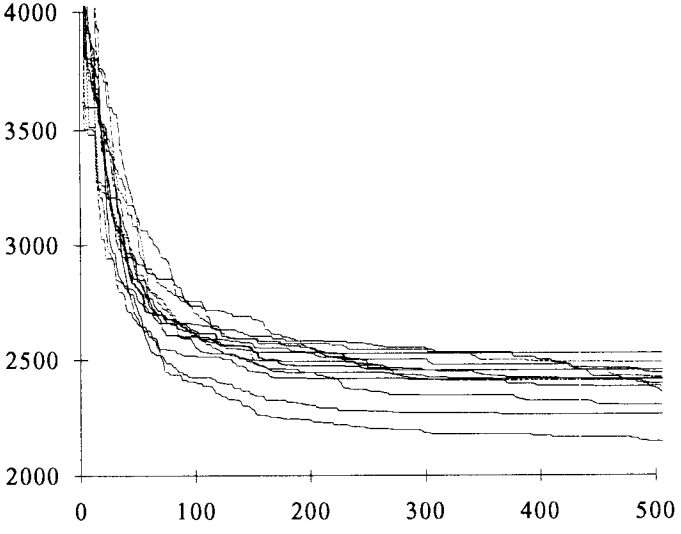

Fig. 2: exponential decreasing schedule.

DEGREE OF PARALLELISM

Running independent SA processes on different processors is a strategy that obviously gives an efficiency equal to 1. The execution time is in fact perfectly divided by the number of processors employed. So that, in order to achieve the highest efficiency of the computational system, it is important to determine the minimum number of processes that gives the higher probability of including the best solution.

When tuning the sequential application, we set the values of the implementation parameters (initial value of the control parameter, decreasing schedule, number of iterations for each level of the control parameter) so that each individual process has the same probability of reaching the optimum solution within the same number of iterations.

However, if we decide to perform a high number of executions (for instance 128), we can observe that only some of them reach values of a given precision.

The ratio between this number of processes and the previous quantity can be considered as the efficiency of the whole set of computations when referred to a given precision.

We can relate its value to the complexity of the specific problem and to the settings of the above parameters; however, at present, we have no general rule for determining this value. It can be done only empirically.

To this end, we observed the distances between the best solutions obtained by 128 executions of SA. In Fig 3, the distribution of the minimum risk is shown for a portfolio selection of 5 shares and 5 periods.

As can be seen, on average, the are six solutions having values close to 1200; this means that, if we want to get solutions falling near that ring, we should run the SA 21 at least times (128/6). Hence, this number can be hence considered the degree of parallelism we are looking for.

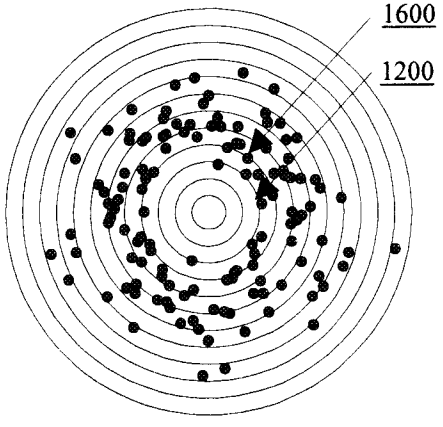

Fig. 3 : portfolio with 5 shares and 5 periods.

In fig. 4 and 5, the same types of distribution are shown: one for a problem of 5 shares and 10 periods, and the other of 15 shares and 10 periods. In both cases, there are two solutions close to the inner ring; therefore we should run the SA at least 64 times (128/2).

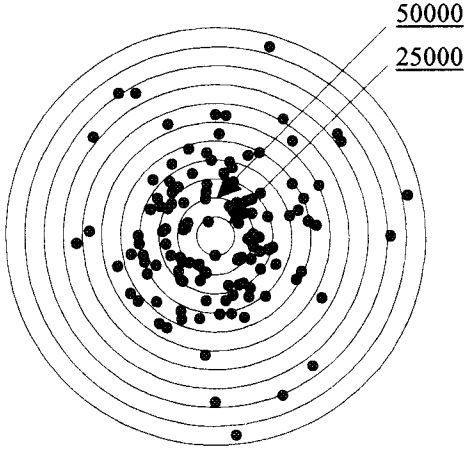

Fig.4 : portfolio with 5 shares and 10 periods.

As expected, the complexity of the problem has a determining influence on the degree of parallelism, but it is important to underline that these values are only

indicative. A better indication is given by the distance between the best solutions and the next best solutions.
For the first case (5 x 5), we have six solutions very close to the minimum value (1200). However the distance between them and the next nearest other ones is not so significant, so we can extend the range of the acceptable solutions.
For instance, if we are satisfied with a level of the objective function up to 1600, we find that the degree of parallelism gradually becomes about 5 (128/25).
Differently, observing the chart in Fig. 4, which concerns the case (5 x 10), we note that there are two best solutions close to the 25000 level, but there are no neighbouring solutions. In this case it is a better option to maintain the degree of parallelism at its upper value (64) at least.
In the last case, we again have two solutions close to the minimum level (10^7); however there are some other neighbouring solutions. In this case we can reduce the degree of parallelism, for instance to 25 (128/5).

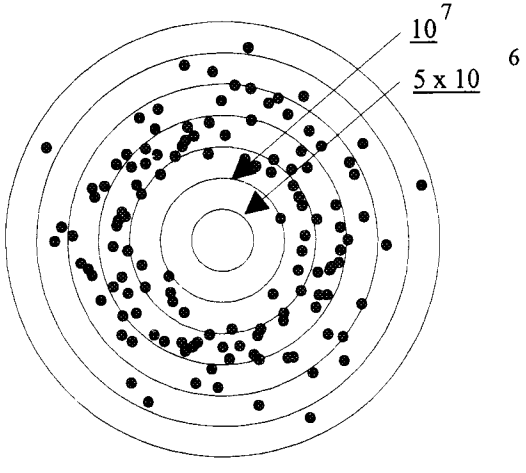

Fig. 5 : portfolio with 15 shares and 10 periods.

Thus, the criterion based on the distance between solutions can offer suggestions about the most suitable degree of parallelism, more effective than the ones based on the size of the problems.
Furthermore, when such a reduction of the degree of parallelism can takes place, one can also rely on the improvements that an actual parallel environment can create. As we will see, applying suitable strategies to the co-operation between processes, it is possible to reach better solutions, thus increasing the efficiency of the computational system.

CO-OPERATING PROCESSES

The exponential decrease of the control parameter, as shown in Fig. 2, determines the differentiation between the trajectories so that the most promising

ones can be identified. This allows us to predict that there are a number of processes that will continue running on paths that stand practically no chance of reaching good approximations to the solution.
We could consider this situation as a redundancy that provokes downgrading of the efficiency. Therefore we can consider the option of re-addressing the processes giving unpromising trajectories. They can be more conveniently utilised to speedup the search drawn by the most promising trajectories.
There are many avenues that one can follow to make the processes co-operate with one another to this end.The most important question to solve is the criterion for deciding when a process can be considered unpromising. We must take into account that when the control parameter has a high value, the co-operation between processes involved in building the same Markow chain has low efficiency. Almost all the configuration are accepted, but only one can be included in the chain.Therefore, it is a better option to let the processes initially follow individual paths into the space of solutions. At selected stages, the process with the highest result can be considered unpromising and therefore it can be forced to restart from the best solution found by one of the other processes.We must also consider that if the co-operation provokes too fast a decrease of the objective function, the probability of a long staying into the area of a local minimum increases. This consideration suggests that the intervention should not take place too frequently. It will also stop the process from leaving a useful path too early.We experimented with many strategies. Among them, the one that gave the best results consists in counting the number of iterations in which a process does not decrease its best intermediate solution. When a process holding the highest value exceeds a given threshold in such a condition, it is forced to sprout a new branch from the best global intermediate solution at that moment.
Fig. 6 shows the results given by different values of this threshold. Each unit consists in a burst of iterations performed at the same level of the control parameter.

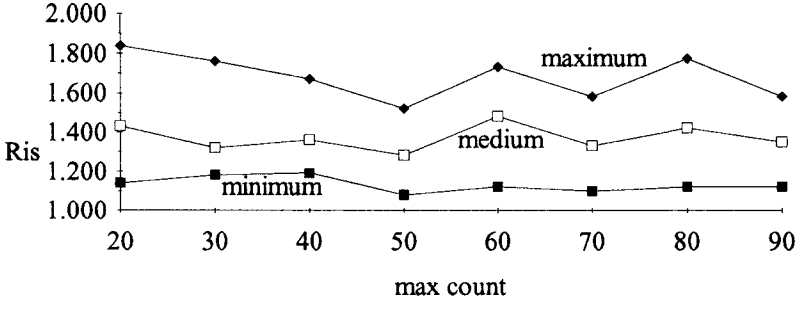

Fig. 6 : threshold selection.

The best value for the examined problem (5 x 5) is 50. It means that the process persisting in the highest value of the intermediate solution for more than 50

reduction of the control parameter, must be considered unpromising and, hence, re-addressed to sprout a new branch.
Fig. 7 shows the behaviour of the trajectories when adopting this strategy and Fig. 8 gives the plots of 32+32 executions of 4 concurrent processes.

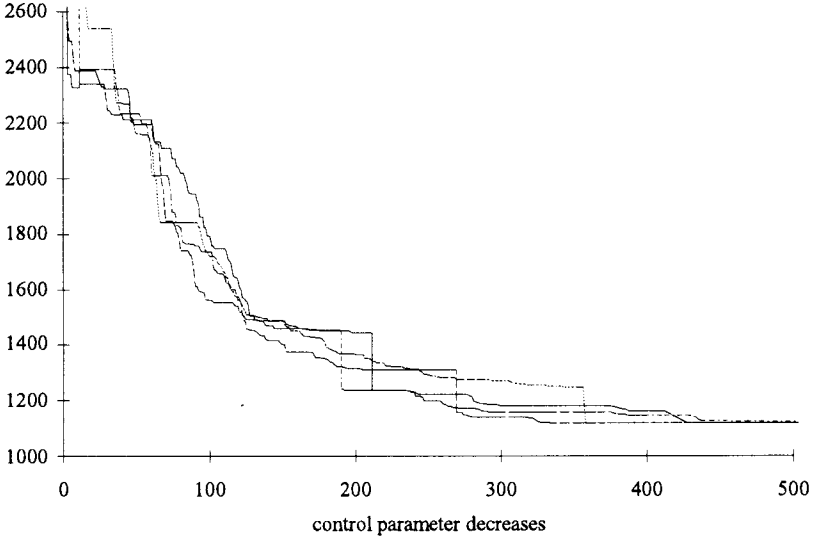

Fig. 7 : four co-operating processes.

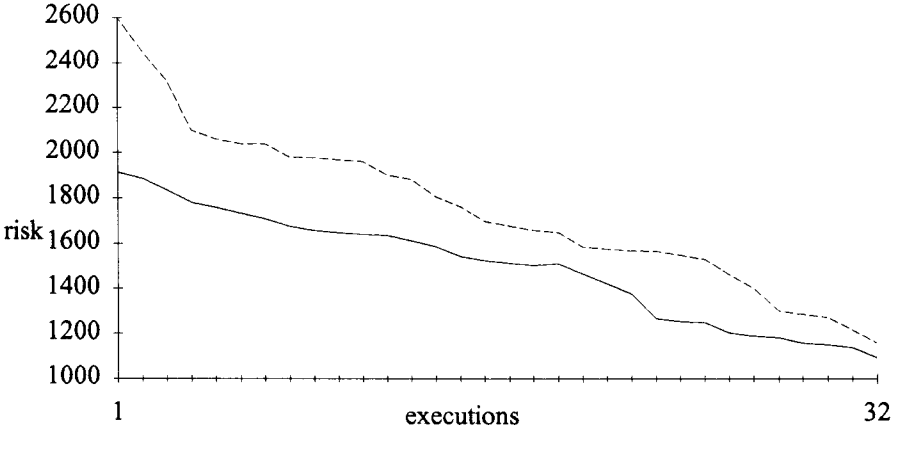

Fig. 8 : independent and co-operating executions

The upper curve concerns independent processes, while the lower one draws the results obtained by 4 co-operating processes.

1911	2597
1887	2453
1837	2320
1781	2099
1758	2060
1732	2039
1708	2038
1675	1979
1657	1976
1647	1966
1640	1960
1635	1900
1610	1882
1585	1806
1543	1761
1524	1696
1512	1676
1509	1656
1500	1646
1463	1582
1421	1573
1375	1566
1266	1563
1253	1545
1248	1529
1202	1460
1187	1399
1181	1298
1156	1283
1149	1269
1137	1216
1094	1156

Tab. 1

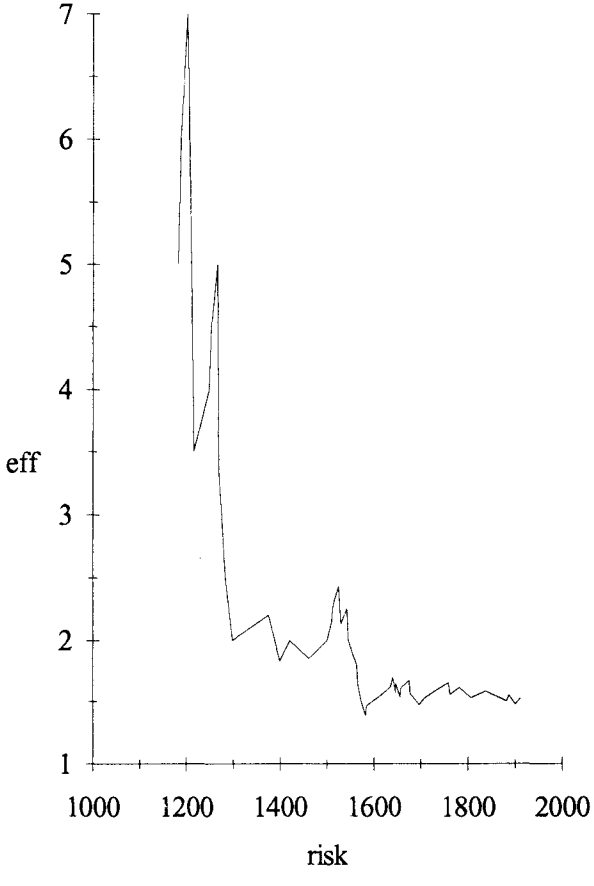

Fig. 9 : efficiency of the co-operation strategy

The results were previously arranged in descending order to better highlight the differences. The same values are listed in Tab.1 where the column on the left reports the best results of each group of four co-operating processes and the other column lists the results of four independent processes.

Finally, Fig.9 plots the efficiency ratios between the two columns of results. For instance, the value 1269 gives an efficiency ratio of 3.3 because this value is reached by 3 runs of four independent processes and by 10 runs of four co-operating processes. Therefore the efficiency is 10/3.

Since the efficiency of 4 independent processes is always 1, this ratio can be considered the parallelism efficiency of the co-operation strategy for the given value of the solution.

CONCLUSION

This paper explored the parallel implementation of SA on a multi-period portfolio selection problem with variables constrained to integer values.
Together with an improvement in the level of risk for the selected policies, we achieved a super-linear speedup of the application. The strategy was tailored for distributed memory systems since a reduced amount of communications are performed. A new efficiency evaluation criterion has been introduced, taking into account the number of times a series of executions reaches a given precision of the objective function.

ACKNOWLEDGEMENT

This work has been supported by "Ministero dell'Università e della Ricerca Scientifica e Tecnologica" of Italy

REFERENCES

[1] H. M. Markowitz, Portfolio Selection: efficient diversification of investments, Wiley & Sons, New York, 1959 .
[2] J. K. Sengupta, Optimal portfolio investment in a dynamic horizon, International Journal of Systems Science,Vol. 14, No 7, 789-800,1983.
[3] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Optimization by Simulated Annealing, Science, No 220, 671-680, 1983.
[4] N. Metropolis, A. Rosenbluth, M. Rosenbluth, M. Teller, E. Teller, Equation of state calculation by fast computing machines, Journal of Chemical Physics, Vol. 21, 1087-1091, 1953.
[5] P.J.M. van Laarhoven, E.H. Aarts, Simulated Annealing: Theory and Application, Kluver Academic Publishers, Dordrecht, 1989.
[6] P.J.M. van Laarhoven, E.H.L. Aarts, J.K. Lenstra, Job Shop Scheduling by Simulated Annealing, Operations Research, Vol. 41, No. 1, 113-125, 1992.
[7] E. Aarts, J Korst, Simulated Annealing and Boltzmann Machines, John Wiley & Sons, New York, 1990.
[8] R. Azencott, Simulated Annealing: Parallelization Techniques, John Wiley & Sons,New York, 1992.
[9] D. Abramson, Constructing School Timetables Using Simulated Annealing: Sequential and Parallel Algorithms, Management Science, Vol. 37, No. 1, 98-113, 1991.
[10] L. Brochard, Efficiency of Some Parallel Numerical Algorithms on Distributed Systems, Parallel Computing, Vol. 12, 21-44, North Holland, 1989.
[11] D. Fischer, On Superlinear speedups, Parallel Computing 17 695- 697, North Holland, 1991
[12] A. Genco Parallel Performance Prediction of Time and Space Scalable Problems, Proc of the European Workshop on Parallel Computing (EWPC'92) - Barcellona (Sp), IOS Press: Parallel Computing: From Teory to Sound Practice, 160-163, 1992