A Hierarchical Approach in Distributed Evolutionary Algorithms for Multiobjective Optimization

Daniela Zaharie, Dana Petcu, and Silviu Panica

Department of Computer Science, West University of Timişoara,
and Institute e-Austria Timişoara
bv. V. Pârvan, no. 4, 300223 Timişoara, Romania
dzaharie@info.uvt.ro, petcu@info.uvt.ro, silviu@info.uvt.ro

Abstract. This paper presents a hierarchical and easy configurable framework for the implementation of distributed evolutionary algorithms for multiobjective optimization problems. The proposed approach is based on a layered structure corresponding to different execution environments like single computers, computing clusters and grid infrastructures. Two case studies, one based on a classical test suite in multiobjective optimization and one based on a data mining task, are presented and the results obtained both on a local cluster of computers and in a grid environment illustrates the characteristics of the proposed implementation framework.

1 Introduction

Evolutionary algorithms proved to be adequate metaheuristics in solving multiobjective optimization problems. However, for complex problems characterized by a large number of decision variables and/or objective functions they need large populations and a lot of iterations in order to obtain a good approximation of the Pareto optimal set. In order to solve this problem, different variants for parallelizing and distributing multiobjective evolutionary algorithms (MOEAs) have been proposed in the last years [3,4,7]. Choosing the appropriate variant for a particular problem is a difficult task, thus simultaneously applying different variants and combining their results could be beneficial. The huge computational power offered today by grid infrastructures allows the use of such strategies which could be beneficial especially when the human knowledge on the problem to be solved or on the method to be applied is lacunar.

The approach proposed in this paper is developed in order to be used either on a cluster or in a grid environment and is based on the idea of using one or several colonies of populations. Each colony consists of a set of populations and can be characterized by its own strategies for assigning a search subspace to each population and for ensuring the communication between populations. The results obtained by all colonies are to be collected and combined in order to obtain the global approximation of the Pareto optimal set and/or Pareto front.

I. Lirkov, S. Margenov, and J. Waśniewski (Eds.): LSSC 2007, LNCS 4818, pp. 516–523, 2008.

The paper is organized as follows. Section 2 presents a brief overview of existing distributed variants of MOEAs. The hierarchical approach and the particularities of the cluster and grid layers are presented in Section 3. In Section 4 two case studies are presented, one involving a classical test suite in MOEAs analysis, and the other one related to a data mining task, the problem of attributes' selection.

2 Distributed Versions of Multiobjective Evolutionary Algorithms

Most evolutionary algorithms for multi-objective optimization use a population of elements which are transformed by recombination and mutation during a given number of generations. At each generation all objective functions are evaluated for all elements of the population and the non-dominance relationship between them is analyzed. In the case of a minimization problem involving r objective functions, f_1,...,f_r an element $x \in D \subset R^n$ is considered non-dominated if there does not exist another element $y \in D$ such that $f_i(y) \leq f_i(x)$ for all $i \in \{1, ..., r\}$ and the inequality is strict for at least one function. The non-dominated elements of the population represent an approximation of the Pareto optimal set. Both the evaluation of elements and the analysis of the nondominance relationship are high cost operations. These costs can be reduced by dividing the population in subpopulations and by evolving them in parallel. In order to design such a distributed variant of a MOEA some key issues should be addressed: the division of the search space, the communication between subpopulations and the combination of the results obtained by all subpopulations.

The division of the search space can be made before starting the evolution (apriori division rule) or dynamically during the evolution (dynamic division rule). In the last years different strategies have been proposed, most of them being based on dynamic division rules [4,7]. Dynamic division rules usually involve some operations aiming to periodically reorganize the structure of the search space. In some cases this operations could be costly by themselves, as for instance in [7], where a clustering step involving the elements of all subpopulations is executed, or in [4], where all subpopulations are gathered and their elements are sorted. On the other hand, applying apriori division rules (e.g. dividing the decision variables space in disjoint or overlapping regions) does not usually involve supplementary costs.

The communication between subpopulations plays a critical role in the behavior of a distributed MOEA. The main components of a communication process are: the communication topology, the communication policy and the communication parameters. The communication topology defines the relationship between subpopulations and the most common variants are: fully connected topology (each subpopulation is allowed to communicate with any other subpopulation) and neighborhood based topologies like the ring topology (a subpopulation S_i can communicate only with subpopulations S_{i-1} and S_{i+1}, in a circular manner).

The communication policy refers to the manner the migrants are selected from the source subpopulation and the way they are assimilated into the destination subpopulation. The migrants can be randomly selected from the entire population or just from the non-dominated subset (elitist selection). The assimilation of the migrants can be realized by just replacing an element of the target subpopulation with the immigrant (the so-called pollination [3]) or by sending back an element to the source subpopulation in order to replace the emigrant (plain migration [9]). The communication topologies and policies can be combined in different manners leading to a large number of strategies.

The parameters influencing the behavior of the distributed variant are: the communication frequency (number of generations between consecutive migration steps) and the migration probability (the probability of an element to be selected for migration).

Besides the large number of parallel implementations of MOEAs, grid implementations have been also recently reported. In [6] is proposed a Globus based implementation of a Pareto archived evolution strategy characterized by remotely executing a number of sequential algorithms on different grid machines and storing the approximated fronts which satisfy some quality criteria. In [5] is presented a grid-enabled framework which allows the design and deployment of parallel hybrid meta-heuristic, including evolutionary algorithms.

3 The Hierarchical Approach

The existence of different MOEAs distribution strategies on one hand and of different architectures on which such algorithms can be executed, on the other hand, motivated us to search for an easy configurable framework for the execution of distributed MOEAs. The approach we propose is based on a layered structure, as illustrated in Figure 1, allowing the execution either on a single computer, on a cluster of computers or in a grid infrastructure.

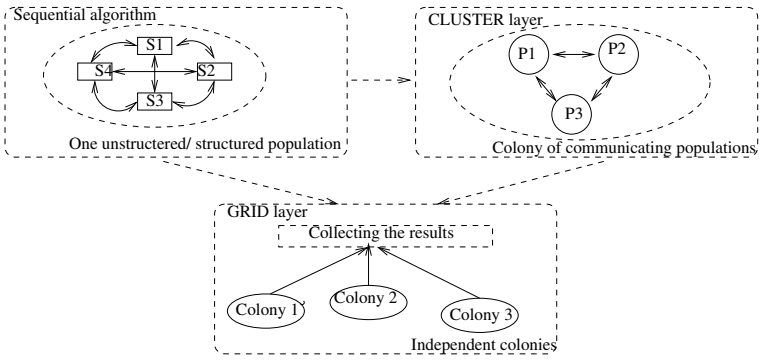

Fig. 1. The layered structure corresponding to the hierarchical approach

The first layer corresponds to the evolution of either a single unstructured population which explores the entire decision space or to a structured population consisting of communicating subpopulations. From an implementation point of view this would correspond to a sequential implementation on one processor. Using a structured population could be beneficial by increasing the population diversity and reducing the cost of the selection step even in the case of a sequential implementation.

The second layer corresponds to a colony of populations which evolve independently but which periodically change some information during a migration step. From an implementation point of view this layer would correspond to a parallel variant executed on a cluster of computers (each processor deals with the evolution of one or several populations from the colony).

The third layer corresponds to the evolution of several independent colonies of populations each one being executed in a location of the grid environment. Unlike the populations in a colony, the colonies are loosely coupled in order to deal with the heterogeneous character of the grid infrastructure. In the current implementation the only communication between the colonies is in the end of the evolution process where the results are collected from all of them. Since the same problem is solved on each colony this induce a certain level of redundancy. In order to reduce the redundancy, each colony can be based on a different MOEA and on different distribution strategies.

The cost of communication between (sub)populations is highly dependent on the layer, thus the communication strategy should be adequately chosen for each layer. There are two main communication processes involved: a periodical communication corresponding to migration stages and a final communication step corresponding to the collection of the partial results obtained by all processes.

Since the aim of the final communication step is just to collect the results, a natural way to implement it is: all processes send their results to a master process which collect them and construct the final result. These partial results are sets of non-dominated elements having almost the same number of elements as the population. If the final results produced by p populations of size m are collected through a message passing interface, the cost of this communication is $\mathcal{O}(pmL)$, L being the size of each element (which depends on the number of decision variables and objective functions).

In the first layer an intensive periodical communication between subpopulations can be applied, including strategies based on gathering and redistributing the entire population. If the evolution of a colony of populations is executed in parallel on a cluster of computers there could be different approaches in implementing the periodical communication when using a message passing interface. Let us consider the case of a colony consisting of p populations. For instance, in the case of random pollination (each population sends some elements to other randomly selected populations), a direct implementation strategy could lead to a number of messages of $\mathcal{O}(p^2)$ to be transferred between p processes. In the case of plain migration, when for each immigrant a replacing element is sent back to the source population the number of messages is twice as in the case of pollination.

The number of messages can be significantly reduced ($\mathcal{O}(p)$) if all processes send their migrants to a master process which distribute them to the target processes. If each message containing migrants is preceded by a short message containing their number, the total number of messages is at most $4(p-1)$. The number of messages transmitted between processors is even smaller in the case of ring topologies ($2p$). The length of messages containing migrants depends on the size of each element, L, on the population size, m, and on the migration probability, p_m. The averaged length of the messages containing migrants is $mp_mL(p-1)/p$.

For the grid layer at least two scenarios can be identified: (i) in each grid location a sequential job corresponding to one colony is executed; (ii) a parallel job involving a message passing interface in the grid infrastructure is executed. In the first case there is no direct communication between colonies, the jobs launched in the grid environment are independently executed (as in [6]) and they send their results through files transfer to the location which initiated the jobs. In the second case low frequency periodical migration should be applied between colonies in order to limit the communication between different sites.

4 Experimental Results and Discussion

The experiments were conducted for the first from the above mentioned scenarios and the behavior of the proposed approach was tested in two distinct contexts: (i) one problem — several strategies; (ii) one strategy — multiple subproblems (e.g. data subsets).

4.1 Case Studies

In the first case we used the test suite from [10], characterized by two objective functions, and we applied different MOEAs (e.g. Nondominated Sorting Genetic Algorithm [1] and Pareto Differential Evolution [9]) with different parameters and distribution strategies for different colonies. Both algorithms use similar selection operators based on computing non-domination ranks and crowding factors.

The second case study is related to the problem of attributes subset selection which consists in identifying, starting from a training set, the most relevant attributes. Such a problem can be solved by assigning to attributes some weights which optimize three criteria [8]: intra-class distance (to be minimized), inter-class distance (to be maximized) and an attribute-class correlation measure (to be maximized). Interpreting this optimization problem as a multiobjective one, the result will be a set of attributes weights, each one leading to a ranking of attributes, where the first attributes are the most relevant ones. The final ranking can be obtained by averaging the rankings corresponding to all elements of the approximated Pareto set. Since the estimation of the attribute-class correlation measure is quadratic with respect to the number of elements in the training set, the evaluation of each element of the population is costly. A natural approach is

to split, by using a proportional sampling strategy, the data set in smaller subsets, and to apply a MOEA independently for each subset. The results obtained for all subsets are combined in order to construct the final ranking.

4.2 Results in a Cluster Environment

The tests corresponding to the cluster layer were based on a parallel implementation using mpiJava and were conducted on a local heterogeneous cluster of 8 nodes (Intel P4, 6 CPUs at 3.0 GHz and 2 CPUs at 2.4 GHz) connected through optical fiber and a Myricom switch at 2 Gb/s. The evolutionary process involved a colony of c populations to be executed on p processors. Since $c \geq p$ each processor deals with a subcolony of c/p populations. Therefore, different communication strategies can be applied between the populations in the subcolony assigned to one processor and between populations assigned to different processors. Figure 2 illustrates the influence of the communication strategy and that of the problem complexity on the speedup ratio in two cases: when the time needed for collecting the results is ignored and when this final communication time is taken into account. The reported results were obtained in the case of 24 populations each one having 20 elements which evolve for 250 generations by using a NSGA-II algorithm [1] and communicate every 25 generations. It follows that for simple test problems (e.g. ZDT2 from [10] with $n = 100$) the cost of the final communication step (involving long messages) is significant with respect to the cost of other steps, leading to low speedup ratios, while for real problems (e.g. attribute selection in the case of a set of real medical data consisting of 177 instances, each one with $n = 14$ attributes) the final communication cost does not significantly alter the speedup ratio. The decrease in the speedup ratio when 8 processors were used is generated by the heterogeneous character of the processors.

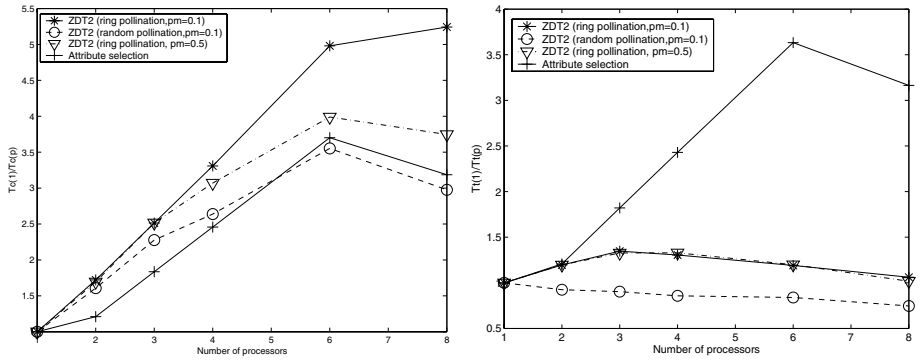

Fig. 2. Speedup ratios when the final communication time is ignored (left) and when the final communication time is taken into consideration (right)

4.3 Results in a Grid Environment

The experiments concerning the grid layer were conducted on the European SEE-GRID infrastructure by using up to 24 nodes with Intel P4 at 3.0 GHz, 1 GB RAM and 100 GB HDD. The tests consisted in launching several sequential and parallel jobs corresponding to different instances of MOEAs. The code was ported on remote sites using gLite. Each MOEA instance is described in a user configuration file specified in the job description. The results generated by different jobs at different sites are transferred through files back to the site which launched the jobs.

The first case study involved 24 sequential jobs corresponding to 24 variants based on two MOEAs, four communication strategies, four variants of search space division and some different values of the specific parameters. All Pareto fronts were compared by using the coverage ratio measure [2] which allows identifying the best result (in our example it was the NSGA-II with one population of 200 elements and a recombination probability of 0.9; the worst behavior corresponds to the same strategy but for a recombination probability of 0.2). Besides the tests involving sequential jobs, experiments with parallel codes executed on clusters from the SEE-GRID virtual organization were also conducted. The possibility of using a larger number of processors than that in the local cluster (e.g. 24 instead of 8) led to a significant decrease of the running time of the evolutionary process. This simple case study illustrates the opportunity offered by the computational grid to efficiently conduct experimental designs when we are looking for appropriate strategies for a given problem.

The second case study was related to the attribute selection problem and involved a set of 2000 synthetic data corresponding to two classes and having 10 attributes. First attribute is just the class label, the next five attributes are randomly generated starting from different distributions for the two classes (e.g. random values generated according to the normal distribution with different parameters for the two classes) and the last four attributes are randomly generated from the same distribution for both classes. Thus a correct ranking would be: first attribute, attributes 2–6, attributes 7–10. Three variants were analyzed: the data were uniformly split in 5, 10 and 20 subsets leading to 5,10 and 20 jobs, respectively. The rankings obtained are: (1,3,5,6,2,4,7,8,9,10) in the case of 5 subsets, (1,6,3,5,2,4,8,7,9,10) in the case of 10 subsets and (1,3,6,5,4,2,7,8,9,10) in the case of 20 subsets. All results are in concordance with the generated data. Concerning the quality of the obtained Pareto front the best results were

Table 1. Coverage ratios (CS) corresponding to Pareto fronts for the data set split in 5,10 and 20 subsets and the corresponding average running time of executing the jobs in the grid environment

CS	5 jobs	10 jobs	20 jobs	Average time (s)
5 jobs	0.0	0.0322	0.002	4485.15± 64.77
10 jobs	0.563	0.0	0.031	1041.95± 194.42
20 jobs	0.912	0.764	0.0	374.71± 70.12

obtained by the variant using 20 subsets. This is illustrated by the coverage ratio measures presented in Table 1($CS(F_1, F_2)$ denotes the ratio of the elements in F_2 which are dominated by elements in F_1) .

5 Conclusions

The hierarchical approach in distributing MOEAs leads to an easy configurable framework allowing the execution either on computational clusters or in a grid infrastructure. Two situations when the grid infrastructure can be efficiently exploited were identified: experimental design of evolutionary algorithms when a large set of strategies should be applied to the same problem and distributed attributes selection for large sets of data when one method can be applied to different data subsets.

Acknowledgment. This work is supported by the projects RO-CEEX 95/ 03.10.2005 (GridMOSI — UVT) and RO-CEEX 65/31.07.2006 (SIAPOM — IeAT).

References

1. Deb, K., et al.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation 6(2), 181–197 (2002)
2. Coello, C.A., van Veldhuizen, D.A., Lamont, G.B.: Evolutionary algorithms for solving multi-objective problems. Kluwer Academic Publishers, Dordrecht (2002)
3. Deb, K., Zope, P., Jain, A.: Distributed computing of Pareto-optimal solutions using multi-objective evolutionary algorithms. In: Fonseca, C.M., et al. (eds.) EMO 2003. LNCS, vol. 2632, pp. 535–549. Springer, Heidelberg (2003)
4. Hiroyasu, T., Miki, M., Watanabe, S.: The new model of parallel genetic algorithm in multi-objective optimization problems — divided range multi-objective genetic algorithm. In: Proc. of IEEE Congress on Evolutionary Computation (CEC 2000), vol. 1, pp. 333–340. IEEE Computer Society, Los Alamitos (2000)
5. Melab, N., Cahon, S., Talbi, E.G.: Grid computing for parallel bioinspired algorithms. J. Parallel Distrib. Comput. 66, 1052–1061 (2006)
6. Nebro, A.J., Alba, E., Luna, F.: Observations in using grid technologies for multi-objective optimization. In: Di Martino, B., et al. (eds.) Engineering the Grid, pp. 27–39 (2006)
7. Streichert, F., Ulmer, H., Zell, A.: Parallelization of multi-objective evolutionary algorithms using clustering algorithms. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 92–107. Springer, Heidelberg (2005)
8. Wang, L., Fu, X.: Data Mining with Computational Intelligence. Springer, Berlin (2005)
9. Zaharie, D., Petcu, D.: Adaptive Pareto differential evolution and its parallelization. In: Wyrzykowski, R., et al. (eds.) PPAM 2004. LNCS, vol. 3019, pp. 261–268. Springer, Heidelberg (2004)
10. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms. IEEE Transactions on Evolutionary Computation 8(2), 125–148 (2000)