A Parallel Framework for Multi-objective Evolutionary Optimization

Dipankar Dasgupta, David Becerra, Alex Banceanu, Fernando Nino, James Simien

Abstract— **This work focuses on the development of a parallel framework method to improve the effectiveness and the efficiency of the obtained solutions by Multi-objective Evolutionary Algorithms. Specifically, a parallel architecture based on JavaSpaces technology and an island paradigm model is proposed and tested on two important and complex computational problems: The Protein Structure Prediction and the Task based Navy's Sailor Assignment problems. An experimental framework is developed in order to test the proposed parallel framework. Particularly, the framework is tested using real-world data belonging to the TSAP and PSP problem. Furthermore, new insights are obtained about modeling these problems as parallel Multi-objective Evolutionary Algorithms.**

Keywords: Parallel Evolutionary Computation, Multi-objective Optimization, Protein Structure Prediction, Task based Sailor Assignment Problem.

I. Introduction

Parallel computing refers to the simultaneous computation of instructions to solve a single problem. Parallel computing has been used to model difficult scientific and engineering problems found in the real world. Optimization problems are among the most interesting and challenging, and they are suitable for the use of parallel techniques. Therefore, they have received an increasing amount of attention in the research and industry community in last decades.

Parallelization techniques have always been a focus of research in evolutionary algorithms for single objective optimization because of their population-based nature. As a natural extension, parallelization techniques have also been used for evolutionary multiobjective optimization. Thus, different parallel approaches applied to single objective optimization problems have been widely suited and reported, particularly, their modeling, performance and behavior have been investigated. However, a modest insight has been given to parallel approaches to solve multi-objective optimization problems. Different parallel models have been proposed in the design of multiobjective optimization algorithms, but only a few of the reported parallel approaches are accessible to compare the performance of those implementations. Then, determining the efficiency and effectiveness of parallel implementations is usually difficult.

In the early work by Veldhuizen et al. [19], they presented the benefits of parallelizing a Multi-objective Optimization Evolutionary Algorithm (MOEA). Additionally, some design issues and considerations were explored and analyzed on a real-world application, particularly, the protein folding problem. Also, a performance analysis in a cluster of an MOEA was carried out in [9] and extended by the same authors in [10]. This analysis was developed using benchmark functions to evaluate the algorithms in different ranges of features such as convexity, discreteness, multimodality and uniformity. On the other hand, in [17], Streicher proposed a divide and conquer approach to parallelize MOEAs; this work aimed at improving the speed of convergence beyond a parallel island MOEA with migration. Additionally, a clustering based parallelization scheme for MOEAs was suggested and compared on standard multi-objective test functions. Recently, in [18] a comprehensive overview of parallel approaches for multiobjective optimization was presented.

Some groups or categories of parallel evolutionay algorithms can be defined based on their common features. Specifically, they can be included in one of three categories: master-slave, islands and diffusion models. In the master-slave model the objective function evaluations are distributed among several slave processors, while a master processor executes the MOEA and other overhead functions. In an island model, MOEA populations are relatively isolated from each other but individuals within some particular island occasionally migrate to another one. As in the master-slave model, the diffusion model deals with one conceptual population, except that each processor holds only a few individuals [19]. Clearly, in all these models a different trade-off between exploration and exploitation of the search space is required.

In this paper, a parallel framework using the JavaSpaces technology over an island model as its parallel paradigm is proposed. This framework is tested using two complex NP-hard problems [7], [13]: The task-based sailor assignment problem (TSAP) and the protein structure prediction problem (PSP). The main goal of this work is to study how a parallelization model can improve the performance of an MOEA. Specifically, the focus of the proposed approach is to improve the effectiveness (i.e., how good its reported solutions are) of the algorithms to find the solutions to TSAP and PSP instances. Therefore, the major contribution of the paper is a parallel MOEA framework which produced an improvement in the quality of the solutions. Moreover, it uses simple and flexible mechanisms for distributed computing.

The outline of the paper is as follows: Section II provides some general information about the use of MOEAs to solve the TSAP and the PSP problem. Then, the proposed parallel

Dipankar Dasgupta, Alex Banceanu are with the Intelligent Security Systems Research Laboratory (ISSRL), University of Memphis, USA. (email: ddasgupt@memphis.edu,alexbanceanu@gmail.com)

David Becerra, Fernando Nino are with the Intelligent Systems Research Laboratory (LISI), National University of Colombia (email:dcbecerrar@unal.edu.co, lfninov@unal.edu.co)

James Simien is with the Navy Personnel Research, Studies, and Technology, Millington, TN, USA. (email: james.simien@navy.mil)

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

computer framework is described in Section III, followed by the experimental framework in Section IV. Some concluding remarks are presented in the last section.

II. Problems

A. Task-based Sailor Assignment Problem (TSAP)

The TSAP is specifically designed for naval bases of the United States Navy, where roughly every three years sailors serving on active duty are reassigned to a different job. Sailors must be reassigned in such a way as to satisfy an array of Naval regulations and personal preferences. TSAP is an inherently multi-objective problem which is a "fine-grained" and more complex version of the sailor assignment problem (SAP). In TSAP, sailors need to be assigned to different jobs (tasks) in different time slots, which are distributed in a certain number of time frames (see Figure 1).

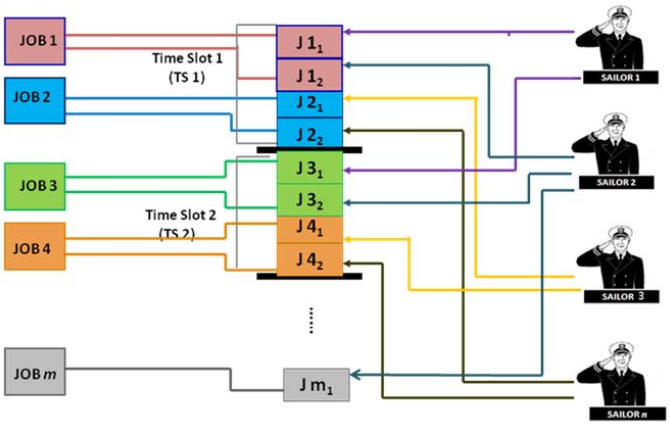

Fig. 1. Graphical description of the TSAP

Let cap_i be the capacity of sailor i, R_{ij} be the resources and C_{ij} be the cost of assigning sailor i to perform task j. Let y_{jk} be the requirement of the number of class j tasks in time slot k, and x_{ijk} be an indicator of whether the sailor i performs task class j in timeslot k or not.

Then the TSAP can be expressed as follows:

• **Decision Space** [decision variables]. A linear chromosome representation is used as follows: the week is divided into a number of days d which are subdivided into k time slots. Each time slot contains a variable number of tasks to be done by eligible sailors. Figure 2 shows the representation of chromosome where each day is divided into multiple time slots and tasks contained in each time slot are done by the eligible sailors; here, $t_{m,j}$ represents the task class m and its j-th instance to be done by sailor S_i.

• **Objective space** [objective functions]. The objective functions of this entire problem are the following:

1) Minimize the number of sailors assigned to navy tasks
2) Minimize the sailor-task assignment cost
3) Maximize the training score (TS) of a particular sailor
4) Maximize the sailor preference (SR) of specific tasks
5) Maximize the commander preference (CR)

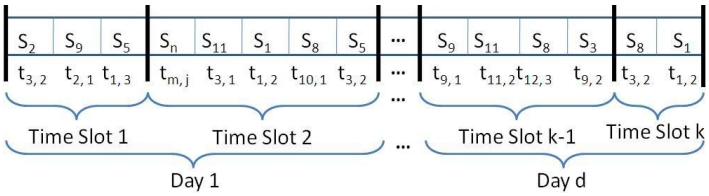

Fig. 2. Chromosome representation for TSAP

Formally, $\mathbf{x} = (x_{ijk})_{i,j,k}$; f_{obj} for $obj = 1, ..., N$ are the objectives to be minimized defined as

$$f_{obj}(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{t} x_{ijk} C_{ij}^{obj}$$

with C_{ij}^{obj} the cost of assigning sailor i to task j on time slot k determined by objective obj;

• **Feasible regions** [equality/inequality constraints] The TSAP is subject to the following constraints:

1) $\sum_{j=1}^{m} x_{ijk} R_{ij} \leq cap_i, \forall i, \forall k$
2) $\sum_{j=1}^{m} x_{ijk} \leq 1, \forall i, \forall k$
3) $\sum_{i=1}^{n} x_{ijk} = y_{jk}, \forall j, \forall k$

with $x_{ijk} \in \{0, 1\}, i \in \{1, .., n\}, j \in \{1, .., m\}, k \in \{1, .., t\}$

B. The Protein Structure Prediction Problem (PSP)

The protein structure prediction problem is one of the most important and challenging open interdisciplinary problems and it consists of determining the folded protein structure from its amino acid sequence. In other words, it is the task of predicting how the information coded in an amino acid sequence of proteins translates into the three-dimensional structure of a biologically active protein (see Figure 3).

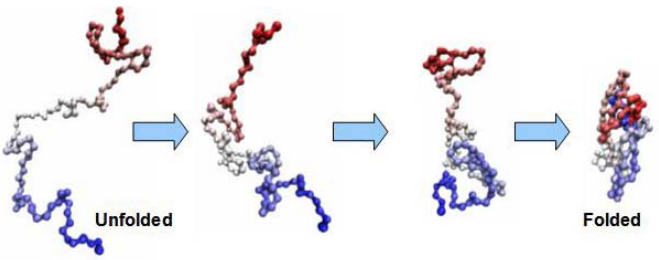

Fig. 3. Graphical description of the PSP

A multi-objective formulation has been shown as a suitable methodology to model the PSP problem [5], [8]. In these models, the energy of the protein conformation is minimized. Moreover, some energy functions that model different kinds of interactions can be defined.

• **Decision Space** [decision variables]. In this work and in order to represent the protein conformations, a trigonometric representation was chosen to model and compute the backbone torsion angles (i.e., ϕ and ψ) and the side-chain torsion angles (i.e., χ) of a protein. Additionally, the

backbone torsion angles and side-chain angles are bounded in regions derived from structure prediction and rotamer libraries, respectively (see Figure 4).

• **Objective space** [objective functions].

In the proposed approach, the different terms of the CHARMm energy function were transformed into several objectives to fit the multiobjective evolutionary optimization formulation of the PSP problem (see [3] for a detailed description of CHARMm). Therefore, a problem with two objectives is considered, where the bonds, angles and torsion interactions between atoms (i.e., local interactions) were defined as the first objective. The second objective considers non-local interaction, thus, it consists of all the interactions between all the atoms not connected by a chemical bond, (i.e., atoms separated by at least three covalent bonds). Typically, an atom energy function has the folowing form.

$$E(R) = \sum_{bonds} B(R) + \sum_{angles} A(R) + \sum_{torsions} T(R) + \sum_{non-bonded} N(R) + [others]$$

where R is the vector representing the conformation of the molecule, and $[others]$ refer to specific terms such as hydrogen bonding in biochemical systems.

• **Feasible regions** [equality/inequality constraints] The constraints which represent unfeasible regions in the energy space are the steric constraints. Then, these constraints are modeled based on the cost function and not as optimization constraints.

Fig. 4. Chromosome representation for PSP

III. The Parallel Framework

The proposed parallel framework was implemented using the JavaSpaces technology. JavaSpaces was the chosen high-level coordination tool for gluing processes together into the distributed application because it allowed the development of a simple design and implementation.

In addition, the proposed approach uses an island model as its parallel paradigm. Particularly, a master processor was used for managing tasks and several slave processors to execute the MOEA. Accordingly, the master processor generates tasks, writes them into a space and collects results from the space. Figure 5 depicts the main components of the JavaSpaces technology. In addition, Figure 6 depicts a scheme of the proposed framework. Particularly, three main tasks are performed: initiating work and slave registration, handling migrations, and finishing work and sending and collecting results.

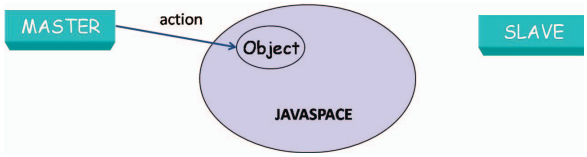

Fig. 5. JavaSpaces technology components

A. Initiating Work and Slave Registration

The proposed implementation requires that the master keeps an updated list of the slaves working for it at any time so it can properly synchronize them. In order to accomplish this, all the slaves must register with their master.

Once a master starts, the first step consists of writing (W) an entry (*InitEntry*) into the JavaSpaces. This entry will then be taken by slaves that can perform the task. This entry is read (R) by the slaves. It is important to note that the entry must only be read (instead of being taken) by the slaves, this is because the entry "Init" must be kept in the JavaSpaces during all the processes so that new slaves can be added at any time.

Subsequently, once a slave receives the *InitEntry*, it requests registration from the master. The slave then writes an unapproved *SlaveEntry*. Then, the master takes this entry and writes the same entry that was taken before. The only difference is that this entry has been already approved by the master.

In the last step, the slave takes this entry, and writes it back; the reason for this rewriting is that the slave must be the author of the entry in the space in order to keep its lease alive. This entry lease is renewed in the background by the slave so that the master can know when the slave is not working on the task anymore, either because it completed the task successfully or due to an unexpected error, such as a network error or unavailability of the entry. At the end of this step, the master has a list of all the slaves that are working for it. The master will periodically check the status of its current slaves; it will also repeat this process to receive new slaves.

B. Handling Migrations

At a given point of the execution of the algorithm a migration will occur, and each of the slaves will send *EmigrantEntries* to the JavaSpaces and then wait for an *ImmigrantEntry* to be sent to it by the master. The master will then collect all *EmigrantEntries* that are sent out by its slaves and will store them. After all slaves have sent their emigrants, the master creates an *ImmigrantEntry* for each one of its slaves based on the emigrant populations that it has collected in the previous step, and it will write all of them to the JavaSpaces. Finally, each slave is able to take its corresponding immigrant and continue the execution of its task, i.e., the MOEA.

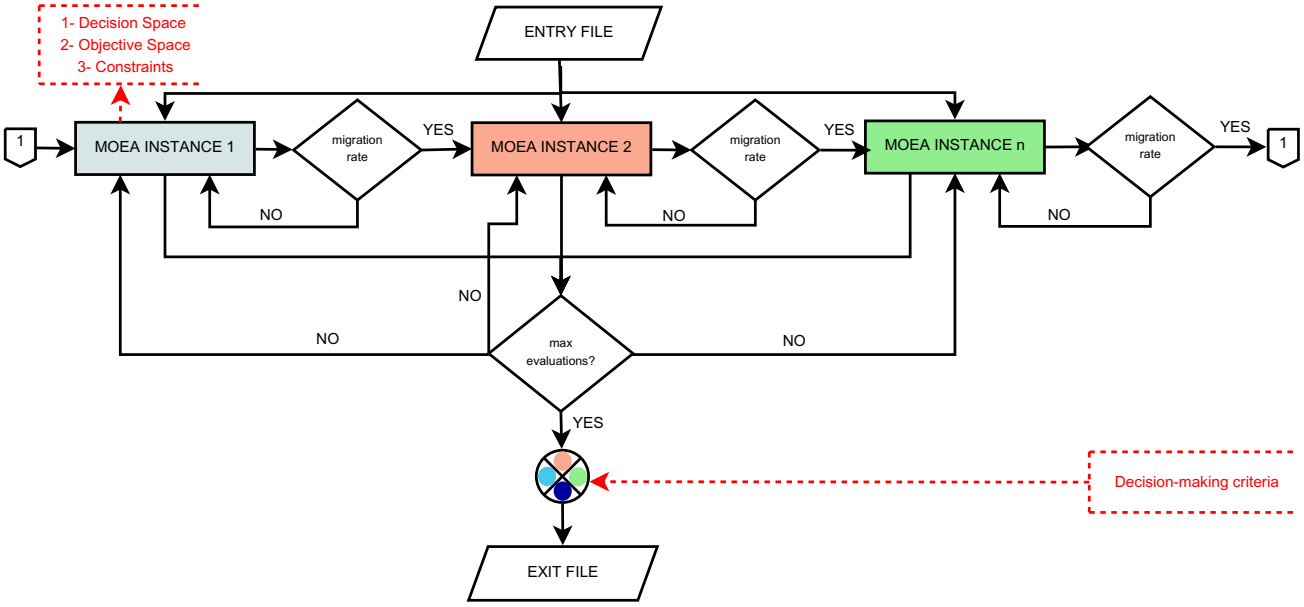

Fig. 6. The proposed parallel framework for multi-objective evolutionary optimization

C. Finishing Work and Sending Results

When a slave has reached the maximum number of evaluations in the MOEA, it writes a *ResultEntry* into the JavaSpaces. These entries are then taken by the master, which adds them into its final result set. When the first *ResultEntry* is taken by the master, it also cancels the *InitEntry*, so that no new slaves begin working on it past this point in time. After this point the master waits for its slave list to empty, either because the slaves complete their work or due to an unexpected error that prevented a slave from finishing its computation. Once this occurs, the master presents its final result set, comprised of all the results received.

IV. Experimental Results

An experimental framework was developed in order to study the impact of the proposed parallel model over the TSAP and PSP problems. Specifically, the experimental framework analyzes different parameters in the island model architecture and the migration policies; it also studies how those parameters affect the MOEA's outcomes. The experiments aim to determine the set of parameters that provide an adequate cover of the search space improving the precision of the model and the quality of the obtained Pareto fronts.

Although island models have been widely studied, there is still no full understanding of the role of the model's basic parameters. Accordingly, it is important to understand what influence each parameter has, how it interacts with the other parameters and how these interactions impact the results of a specific problem.

To thoroughly study the impact of the parallel framework on the behavior of the MOEA, some parameters were fixed and several experiments for all combinations of those parameters were performed. Specifically, to evaluate the impact of the number of islands, a ring topology was used with different numbers of islands. To study the impact of the migration policies different migration rates and different portions of the population to be migrated were used. The values of the parameters are summarized in Table I for the TSAP and the PSP problem.

TABLE I
Experimental parameters for TSAP and PSP

Parameter or Policie	Values(TSAP)	Values(PSP)
Island Topology	Ring	Ring
Number of islands	4 and 8	4
Migration rates (times)	5,10 and 20	20
Pop. to be migrated (%)	10, 20 and 30	10
MOEA	NSGA-II [11]	NSGA-II [11]
Number of evaluations	500000	2000000
Population size	100	100
Number of sailors	1000	-
Number of jobs	500	-
SBX Crossover	$\eta_c = 5, p_c = 0.9$	$\eta_c = 5, p_c = 0.9$
Polynomial Mutation	$\eta_m = 10, p_m = 0.01$	$\eta_m = 10, p_m = 0.01$

It is important to stress that the main focus of the experimental framework was to evaluate the improvement in the diversity of the solutions found by the MOEA. In fact, the time improvement was not studied given that JavaSpaces has shown good speedups for parametric experiments in relation to sequential implementations [16].

Typically it is difficult to compare parallel frameworks due to the algorithmic and implementation aspects. Accordingly, a comparison of the proposed framework with other similar parallel systems was not carried out to the due to the lack of access to other frameworks that implemented the same case studies (PSP and TSAP) .

The TSAP ran over a Dual Intel Xeon Quad core 2.6 GHz CPU; 12 GB Memory; Debian Linux v4.3.2 Operating System; and 1 G Ethernet Communication Network. On the other hand, the PSP problem ran over a work-station

equipped with two Quad Core Intel Xeon Processor X5450 (2.33GHz,2X6M L2,1333), with 32GB FBD Memory.

Since Pareto-optimal solutions for considered instances of TSAP are not known, some of the existing performance metrics (such as *coverage* and *proximity* metrics) cannot be used to evaluate the results of the proposed approach. Therefore, the hypervolume was used as a quality measure, which is strict Pareto compliant. The hypervolume indicator is a metric used in evolutionary multi-objective optimization that measures the volume of the dominated portion of the objective space.

Even if there is not a Pareto-front solution for the PSP problem, the optimal solution (i.e., the native protein conformation) is already known. Therefore, the root mean square deviation (RMSD) was used as a quality measure. The RMSD is a well-known metric to evaluate how similar the predicted protein conformation is to the native one (i.e., structural similarity). When an optimal superposition of the two structures that are compared is computed, the RMSD measures the average distance between their backbones.

A. Parallelization Experimental Results for TSAP

For each specific parameter configuration, five independent runs were performed. For each run, the improvement of the hypervolume with respect to the hypervolume obtained by the sequential algorithm was computed. Figures 7 and 8 report the average of the runs for a four island and an eight island configuration. Figure 9 depicts the dispersion of the improvements. Notice that a configuration is built as the combination of the parameters shown in Table I. For example a configuration labeled as s4.p10.m5 means that a problem with 4 different islands, migrating 10 percent of the population, during 5 times in the 500,000 evaluations was executed.

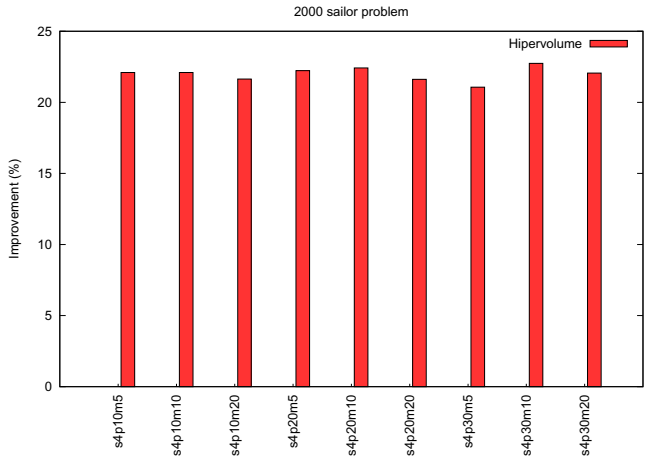

Fig. 8. Average of the improvement of the hypervolume of the parallel conformations with respect to its serial counterpart in the 2000 sailor problem.

In the remaining of this section, a discussion from the experiments (Figures 7, 8 and 9) is presented.

When the number of evaluations is fixed at 500,000, the parallel framework for TSAP provides better solutions than its sequential version, for all the tested configurations. The best performance is obtained by an island of 8 processors migrating 20 times 20 percent of the population. Using this configuration an improvement of 7.11% in the hypervolume measure was obtained with respect to the hypervolume obtained by the sequential counterpart.

In general, the results obtained for all the configurations using 4 islands were outperformed by those using 8 islands.

The obtained improvements in hypervolume measures for the 2000 sailor problem are higher than those of the 1000 sailor problem (See Figures 7 and 8). It is clear that the 2000 sailor problem is a more difficult problem than the 1000 counterpart, and the parallel framework seems to work better in this kind of harder problems. In the 1000 problem the hypervolume value is stabilized before the 500,000 evaluations are exhausted, in contrast, in the two thousand sailors problem this threshold is harder to achieve. Then, the effectiveness of the two thousand sailor problem benefits from the parallel framework because more solutions arise from the simultaneous executions of the TSAP instances spreading the cover of the Pareto front.

However, in the TSAP, the iterations involving a higher number of processors working on it have a beneficial effect as they provide a way to improve the quality of the obtained Pareto fronts. It is not clear if this improvement is significant compared to the additional quantity of computational resources invested in a parallel approach. Serial iterations with the whole population seem to work well in the TSAP for the chosen number of evaluations (500,000), and that makes it unclear if this is the reason for the real improvement of the parallel approach. It has been reported in [6] that in the TSAP sequential algorithm after a certain number of evaluations the hypervolume value will remain almost constant, then more tests should be carried out in order to identify that stabilization point in the parallel approach.

B. Parallelization Experimental Results for the PSP problem

In this section, an analysis of the results obtained over the 1ROP protein is provided. This peptide has been used given that it is one of the most studied proteins both theoretically and experimentally. 1ROP protein is a 63 amino-acid length protein obtained by an experimental X-ray method at a resolution of 1.7 A^o.

TABLE II
COMPUTED PROTEIN CONFORMATION FOR THE 1ROP PROTEIN TARGET

Iter$\times 10^6$	D.M	Err(S)	E(S)	Err (P)	E (P)
0,5	Utility	5.263	-492.43	3.955	-212.95
	Angles	5.367	-325.35	3.980	-225.44
1	Utility	5.323	-605.21	3.915	-476.07
	Angles	5.331	-598.69	3.925	-477.75
1,5	Utility	5.288	-637.47	3.806	-584.00
	Angles	5.327	-624.97	3.801	-583.88
2	Utility	5.296	-666.17	3.826	-656.40
	Angles	5.296	-665.36	3.775	-626.75

Based on a knee-based decision-making method [2], the parallel algorithm matches the crystal structure with an

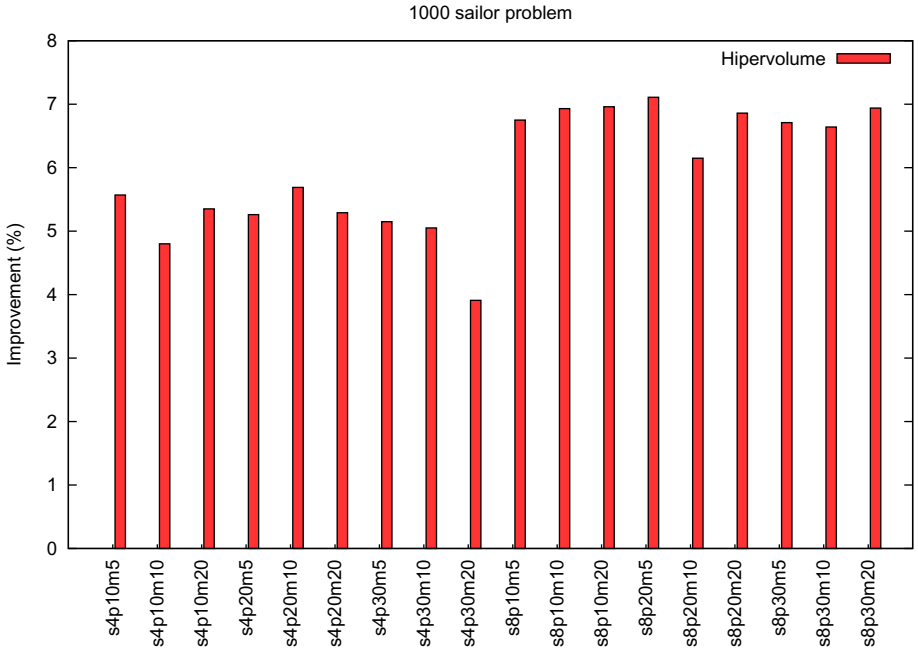

Fig. 7. Average of the improvement of the hypervolume of the parallel conformations with respect to its serial counterpart in the 1000 sailor problem.

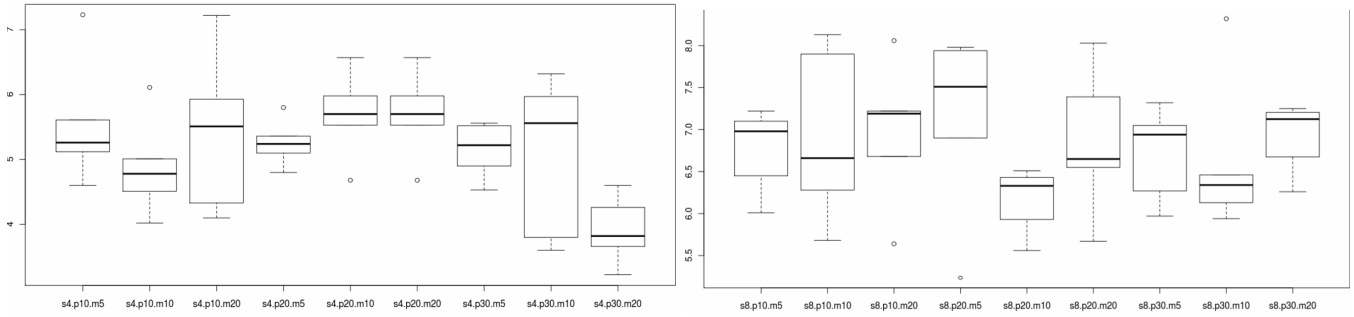

Fig. 9. Box plot diagrams for configurations using 4 and 8 islands.

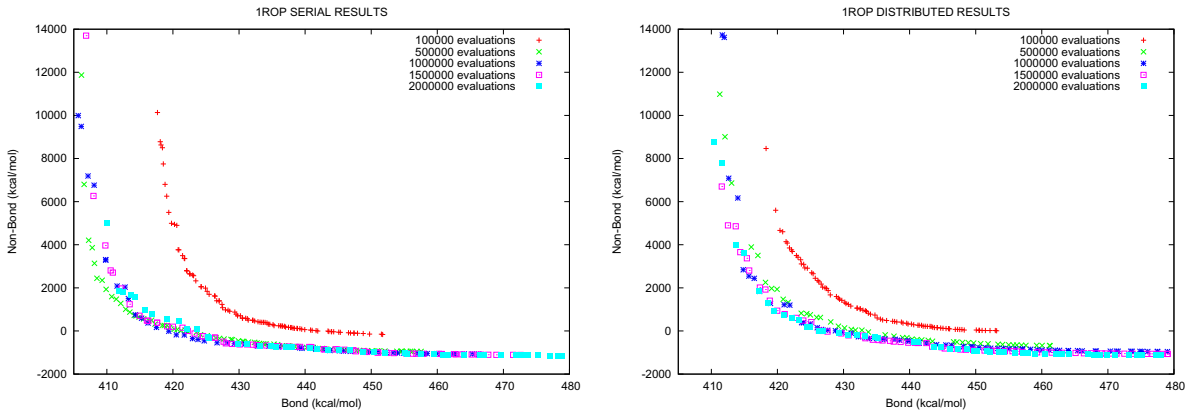

Fig. 10. Pareto Fronts for the serial and parallel algorithm for ROP protein

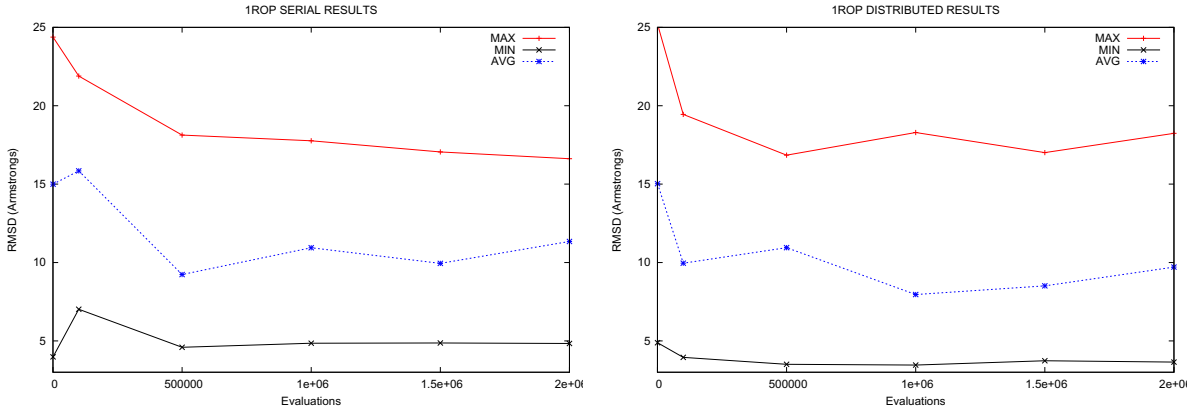

Fig. 11. Histrogram of RMSD ranges (1ROP) for the serial and parallel version of PSP

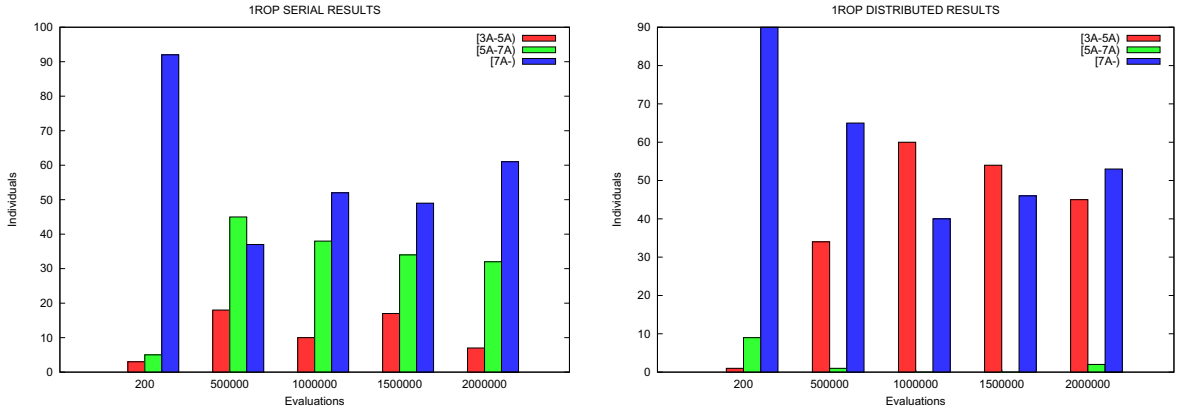

Fig. 12. Histrogram of RMSD ranges (1ROP) for the serial and parallel version of PSP

TABLE III
PROPOSED APPROACH VERSUS OTHER APPROACHES FOR 1ROP

Algorithm	$rmsdC_\alpha(A^o)$
OUR	3.7754
I-PAES [5]	3.70
Scatter [15]	17.25
HC-GA (with hydrophobic term) [4]	5.6
Bhageerath [14]	4.3
CReF [12]	7.1
(1+1)-PAES$_1$ [1]	6.31
(1+1)-PAES$_2$ [1]	8.665

$rmsdC_\alpha$ of 3.7754 A^o and energy -626.75 $kcalmol^{-1}$, meanwhile the serial algorithm matches the crystal structure with an $rmsdC_\alpha$ of 5.2963 A^o and energy -666.17 $kcalmol^{-1}$. Table II reports the computed structures in different iterations of the algorithm.

Figure 10 shows the different Pareto fronts found by the algorithm in different iterations. Figure 11 shows the behavior of each island, with respect to the maximum, minimum and average $rmsdC_\alpha$ found in different stages of the algorithm. Figure 12 shows the frequency histograms of the $rmsdC_\alpha$ errors in specific intervals for the serial and parallel algorithm versions. Note that Figure 12 depicts the amount of individuals belonging to a specific $rmsdC_\alpha$ interval. This measure is highly important because it allows to understand the role of the evolution in the minimization of the RMSD errors without propensities to outlier RMSD values. Finally, Table III reports the comparison of the proposed approach versus other approaches for 1ROP protein. In the following paragraphs, the conclusions from the experiments are discussed.

When the number of evaluations is fixed at two millions, the parallel framework for the PSP problem provides better solutions than its sequential version for the tested configurations (see Figure 11 and Table II). Furthermore, note that the number of protein conformations belonging to the interval [$3A^o$, $5A^o$] is higher for the parallel version than its sequential counterpart. Specifically, in the parallel algorithm this number is close to fifty percent of the population, meanwhile in the serial version it barely reaches seven percent. Figure 12 stresses the importance of the parallel model given that the final collected results maintains the good behavior of the best islands and it is not very sensitive to the behaviors of the worst islands.

It should be stressed that, although the NSGA-II algorithm finds well distributed fronts and it emphasizes the

convergence, there are some individuals in the optimal set that present high energy levels. Typically these individuals show a marked difference in the objective tradeoff. Note that the Pareto front of the parallel algorithm reported in Figure 10 is the set of dominant solutions of joining the Pareto fronts of islands one, two, three and four. It is worth to notice that the serial implementations get lower energy values and higher $rmsdC_\alpha$ errors than their parallel counterparts. This fact suggests that the parallel algorithm performs a better exploration of the energy landscape with respect to its serial implementation. Then, the effectiveness of the PSP implementation benefits from the parallel framework, but this improvement is not clearly perceived.

The iterations involving a higher number of processors working on the PSP problem have a beneficial effect as they provide a way to improve the quality of the obtained protein conformations. Table III reports the comparison of the proposed approach versus other approaches for the 1ROP protein.

V. Conclusions

In this paper, a parallel multi-objective evolutionary computational framework is proposed. This model is evaluated to determine its effectiveness and efficiency of MOEAs. Specifically, two example problems were chosen to evaluate the advantages and disadvantages of the proposed model.

This paper is a contribution to provide a foundation for selecting appropriate computational architectures, associated operators and parameter values. Though problems considered in this work are from two different domains, the proposed parallel framework was able to solve them efficiently. Additionally, this work could help to direct the TSAP and PSP problems toward new unexplored paths.

An improvement of the TSAP results was obtained using a parallel approach. This has been verified by running several experiments over two TSAP instances (1000 and 2000 sailor problems) and changing the combinations of the parameters in the model. Although the standard island TSAP with migration seems to be quite robust and efficient, it is difficult to test and compare the real improvement of the model. Specifically, it is not clear how the migration policies impact the Pareto solutions. Therefore, more experiments need to be performed to conclusively determine how those parameters provide a beneficial effect to increment the diversity of the TSAP solutions.

The proposed parallel framework is a good protein folding predictor. Particularly, an improvement of the PSP results was obtained with respect to its serial counterpart. Additional work may be needed to improve the quality of the energy functions and the policies and parameters used by the model. The proposed approach can certainly benefit from other improvements, such as refinement of the MOEA and the addition of new features including amino acid specific information and biological heuristic information.

Future work will include further case studies, such as the docking problem and some combinatorics problems such as the constrained longest common subsequence problem (CLCS). Based on the proposed parallel approach, several web services are being implemented and will be soon available on-line.

References

[1] M.E. Algorithms. Computational Studies of Peptide and Protein Structure Prediction Problems via Multiobjective Evolutionary Algorithms. *Multiobjective problem solving from nature: from concepts to applications*, page 93, 2008.

[2] J. Branke, K. Deb, H. Dierolf, and M. Osswald. Finding knees in multi-objective optimization. *Lecture Notes in Computer Science*, pages 722–731, 2004.

[3] BR Brooks, RE Bruccoleri, BD Olafson, DJ States, S. Swaminathan, and M. Karplus. Charmm: a program for macromolecular energy, minimization, and dynamics calculations. *Journal of computational chemistry*, 4:187–217, 1983.

[4] LR Cooper, DW Corne, and MJ Crabbe. Use of a novel Hill-climbing genetic algorithm in protein folding simulations. *Computational biology and chemistry*, 27(6):575, 2003.

[5] Vincenzo Cutello, Giuseppe Narzisi, and Giuseppe Nicosia. A multi-objective evolutionary approach to the protein structure prediction problem. *Journal of The Royal Society Interface*, 3:139–151, February 2006.

[6] D. Dasgupta, G. Hernandez, D. Garrett, P. Vejandla, A. Kaushal, R. Yerneni, and J. Simien. A comparison of multiobjective evolutionary algorithms with informed initialization and kuhn-munkres algorithm for the sailor assignment problem. In *Proceedings of the 2008 Genetic and Evolutionary Computation Conference (GECCO-08)*. ACM press, 2008.

[7] D. Dasgupta, F. Nino, D. Garrett, K. Chaudhuri, S. Medapati, A. Kaushal, and J. Simien. A multiobjective evolutionary algorithm for the task based sailor assignment problem. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1475–1482. ACM, 2009.

[8] R.O. Day, J.B. Zydallis, G.B. Lamont, and R. Pachter. Solving the protein structure prediction problem through a multiobjective genetic algorithm. *Nanotechnology*, 2:32–35, 2002.

[9] F. De Toro, J. Ortega, and B. Paechter. Parallel single front genetic algorithm: Performance analysis in a cluster system. In *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, page 8, 2003.

[10] F. de Toro Negro, J. Ortega, E. Ros, S. Mota, B. Paechter, and JM Martın. PSFGA: Parallel processing and evolutionary computation for multiobjective optimisation. *Parallel Computing*, 30(5-6):721–739, 2004.

[11] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *Lecture Notes in Computer Science*, pages 849–858, 2000.

[12] M. Dorn and O.N. de Souza. CReF: a central-residue-fragment-based method for predicting approximate 3-D polypeptides structures. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1261–1267. ACM New York, NY, USA, 2008.

[13] A.S. Fraenkel. Complexity of protein folding. *Bulletin of Mathematical Biology*, 55(6):1199–1210, 1993.

[14] B. Jayaram, K. Bhushan, S.R. Shenoy, P. Narang, S. Bose, P. Agrawal, D. Sahu, and V. Pandey. Bhageerath: an energy based web enabled computer software suite for limiting the search space of tertiary structures of small globular proteins. *Nucleic acids research*, 34(21):6195, 2006.

[15] C. Kehyayan, N. Mansour, and H. Khachfe. Evolutionary Algorithm for Protein Structure Prediction. In *Advanced Computer Theory and Engineering, 2008. ICACTE'08. International Conference on*, pages 925–929, 2008.

[16] M.S. Noble and S. Zlateva. Scientific computation with JavaSpaces. *Lecture Notes in Computer Science*, pages 657–666, 2001.

[17] F. Streichert, H. Ulmer, and A. Zell. Parallelization of multi-objective evolutionary algorithms using clustering algorithms. In *EMO*, pages 92–107. Springer, 2005.

[18] E.G. Talbi, S. Mostaghim, T. Okabe, H. Ishibuchi, G. Rudolph, and C.A.C. Coello. Parallel Approaches for Multiobjective Optimization. *Lecture Notes In Computer Science*, pages 349–372, 2008.

[19] DA Van Veldhuizen, JB Zydallis, and GB Lamont. Considerations in engineering parallel multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7(2):144–173, 2003.