A Homogeneous Distributed Computing Framework for Multi-Objective Evolutionary Algorithm

Ki-Baek Lee and Jong-Hwan Kim, *Fellow, IEEE*,
Department of Electrical Engineering, KAIST,
335 Gwahangno, Yuseong-gu, Daejeon 305-701, Republic of Korea
E-mail: {kblee, johkim}@rit.kaist.ac.kr

Abstract This paper proposes a homogeneous distributed computing (HDC) framework for multi-objective evolutionary algorithm (MOEA). In this framework, multiple processors divide a work into several pieces and carry them out in parallel. Every processor does its task in a homogeneous way so that the overall procedure becomes not only faster but also fault-tolerant and independent to the number of processors. To implement this framework into an evolutionary algorithm, the evolutionary process of multi-objective particle swarm optimization (MOPSO) is employed. The effectiveness of the proposed framework is demonstrated by empirical comparisons between the results with the different numbers of processors, one and four. Seven DTLZ functions are used as benchmark functions and hypervolume, diversity, and evaluation time are used as comparison metrics. The results indicate that the evaluation time is significantly reduced by the proposed framework without any loss of overall solution quality and diversity.

Key words: Distributed computing, Multi-Objective Evolutionary Algorithm, Particle Swarm Optimiztion

1 Introduction

Recently in most of real world problems, the importance of multi-objective evolutionary algorithm (MOEA) has been in the limelight. MOEA is inspired from evolution phenomena of nature and searches for solutions through an evolutionary process. By using MOEA, the parameters of the problems can be efficiently optimized considering multiple objectives simultaneously [1–4]. However, MOEA may require significant computation time as the number of objectives or the number of parameters increases. In this case, some techniques to handle time-consuming tasks can be helpful.

Distributed computing is one of the most promising techniques. Through distributed computing scheme, tasks can be distributed to available processors and processed in parallel. K. Deb *et al.* proposed an approach for finding multiple Pareto-optimal solutions with a distributed computing system [5]. In this approach, each process was in charge of a particular potion of Pareto optimal region for distributed computing. K. C. Tan *et al.* suggested a distributed cooperative coevolutionary algorithm (DCCEA) [6]. To design this algorithm suitable for distributed computing, each parameter was assigned to a subpopulation and these subpopulations were partitioned into groups depending on the number of available peer processors. These studies showed that not only the computation time was reduced but also the diversity of the obtained solution set was improved by incorporating distributed computing within MOEA. However, in spite of the successful results of them, there is still room for improvement. First of all, the algorithms may fail by even one faulty processor because each processor has its unique role and cannot assist each other. In addition, it is inconvenient to adjust the number of processors. Some equations or structures of the algorithms should be modified according to the number of processors. To solve those problem, homogeneous distributed computing framework can be a solution. There are several advantages of homogeneity which means every processor does the same task. Firstly, it makes algorithms more fault-tolerant. Secondly, the number of processors can be readily controlled. Thirdly, there is no need for communication between processors.

Therefore, in this paper, a homogeneous distributed computing (HDC) framework for MOEA is proposed and to implement this framework into an evolutionary algorithm, the evolutionary process of multi-objective particle swarm optimization (MOPSO) is employed. MOPSO is one of the well-known evolutionary algorithms and in particular, it is easy to implement and fast [7]. To demonstrate the effectiveness of the proposed framework, empirical comparison is carried out between the results with the different numbers of processors, one and four. Seven DTLZ functions are used as benchmark functions and hypervolume, diversity, and evaluation time are used as comparison metrics [8–10].

This paper is organized as follows. Section II proposes HDC framework and introduces multi-objective particle swarm optimization with homogeneous distributed computing (MOPSO-HDC), an implementation of it. In Section III, the experimental result of MOPSO-HDC is discussed. Finally, concluding remarks follow in Section IV.

2 Homogeneous Distributed Computing Framework for Multi-Objective Evolutionary Algorithm

In this section, HDC framework for MOEA is described. The distinguishing feature of HDC framework is that multiple processors divide a work into several pieces and carry them out in parallel. In addition, every processor does its task homogeneously and as a result, it is expected that the overall procedure becomes not only

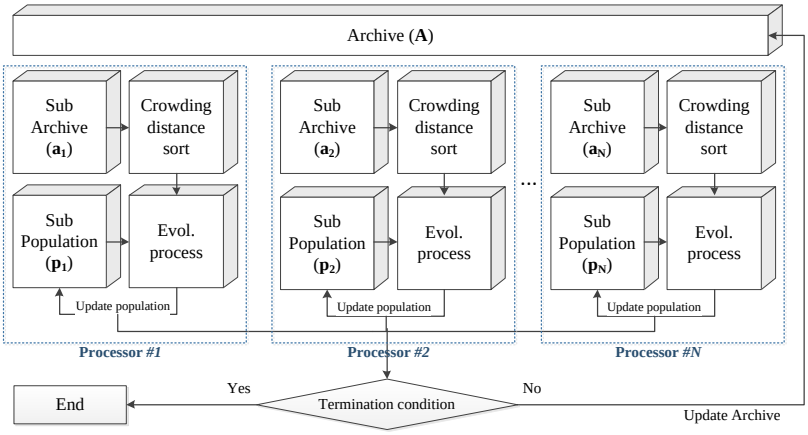

Fig. 1: Flow diagram of HDC framework

faster but also fault tolerant and independent to the number of processors. As an example implementation, multi-objective particle swarm optimization with homogeneous distributed computing (MOPSO-HDC) is introduced. MOPSO is one of the well-known evolutionary algorithms and in particular, it is easy to implement and fast. Thus MOPSO-HDC is a good example for verifying the effectiveness of HDC framework.

2.1 The proposed HDC framework

As categorized in [11], distributed computing can be implimented through three different approaches, i.e., master-slave model, island model, and diffusion model. In the master-slave model, a master processor executes overall MOEA and slave processors handle function evaluations only. In the island model, each processor has there own subpopulation and executes MOEA independently. Some solutions are migrated between the processors with certain frequency as necessary. The diffusion model is a fine-grained variation of the island model. The master-slave model and diffusion model cannot be implemented in homogeneous way. Therefore, in HDC framework, the island model is adopted and the task is homogeneously divided and processed.

Fig. 1 shows the flow diagram of HDC framework. In HDC framework, there are two important sets, population and archive. A population consists of individuals, which search for the solutions by updating themselves through evolutionary process. An archive consists of the non-dominated solution found by the individ-

Algorithm 1 Multi-Objective Particle Swarm Optimization with Homogeneous Distributed Computing

1. Initialize a population and an archive.
2. Divide the task for N available processors.
   - Split the population and the archive equally into N subgroups each.
   - Distribute them to the processors so that each processor has its own sub-population and sub-archive.
3. Update each sub-population.
   for each processor **do**
   Calculate the crowding distances of the solutions in the sub-archive.
   Sort the solutions based on the their crowding distances.
   for each individuals in the sub-population **do**
   Choose the global best position ${}^{g}\mathbf{x}_k^t$ from upper half of the sub-archive.
   Update its velocity and position.
   Evaluate the objective function values.
   Update the personal best position ${}^{p}\mathbf{x}_k^t$.
   end for
   end for
4. Update the global population.
5. Update the global archive.
6. Go back to step 2 and repeat until the termination condition is met.

uals. The individuals in a global population and the non-dominated solutions in a global archive are equally distributed to the sub-population and the sub-archive of each available processor. Each processor evaluates the evolutionary process to update its sub-population with its sub-archive. After every processor finished its update, i.e. synchronization, the global archive is updated by domination test with the updated sub-populations. Finally, the global population is updated by uniting the sub-populations. By repeating these steps until a termination condition is met, desired solutions can be obtained.

2.2 The overall procedure of MOPSO-HDC

The overall Procedure of MOPSO-HDC can be summerized as Algorithm 1 and each step of the algorithm is described in the following.

1. Initialize a population and an archive.

The velocity and position of the individuals in a population are randomly initialized. The velocity $\mathbf{v}_k$ and position $\mathbf{x}_k$ of the k-th individual are the D-dimensional vectors as follows:

$$\mathbf{v}_k \in \mathbb{R}^D, \mathbf{x}_k \in \mathbb{R}^D.$$

The objective function values of each individual are calculated initially and the personal best position of each individual ${}^{p}\mathbf{x}_k^t$ is also set as itself. The archive is initialized as a null set.

2. Divide the task for N available processors.

The population and the archive are equally divided into N sub-populations and N sub-archives. Each processor has its own sub-population and sub-archive.

3. Update each sub-population.

To update an individual in PSO, its personal best position and a global best position are used. The personal best position has been determined from the previous step and the global best position is selected from the sub-archive. To maintain the diversity of the individuals, the crowding distances of the solutions in the sub-archive are calculated and the solutions are sorted by their crowding distances [12]. By selecting the global best position ${}^{g}\mathbf{x}_k^t$ randomly from the upper half of the sub-archive, the individuals can be guided with sufficient diversity. The velocity and position of each individual are updated as follows:

$$\begin{cases} \mathbf{v}_k^t = w \cdot \mathbf{v}_k^{t-1} + c \cdot \{\phi_k^{1,t}({}^{p}\mathbf{x}_k^{t-1} - \mathbf{x}_k^{t-1}) \\ \qquad +\phi_k^{2,t}({}^{g}\mathbf{x}_k^{t} - \mathbf{x}_k^{t-1})\} \\ \mathbf{x}_k^t = \mathbf{x}_k^{t-1} + \mathbf{v}_k^t \end{cases} \tag{1}$$

where w and c are constants and $\phi_k^{1,t}$ and $\phi_k^{2,t}$ are random real values uniformly distributed in $[0, 1]$. $\mathbf{v}_k^t$ and $\mathbf{x}_k^t$ represent the velocity and position of the k-th individual at generation t, respectively. New random values are generated for each individual at each and every generation. After that, objective function values of each particle are evaluated. Finally, the personal best position of each individual ${}^{p}\mathbf{x}_k^t$ is updated.

4. Update the global population.

The global population is updated by uniting the sub-populations.

5. Update the global archive.

The previous global archive and the updated global population is united and dominance test is performed on the union. By discarding dominated solutions, the global archive is updated.

6. Go back to step 2 and repeat until the termination condition is met.

By repeating these steps until the termination condition is met, the desired solutions can be obtained.

Note that computational complexity of the proposed algorithm is governed by the sort process, i.e. the crowding distance-based sort. Since the sort process is done by the quick sort, the algorithm has the average computational complexity of O(nlog(n)).

Table 1: The parameter settings of the algorithm

Parameters	Values
Population size (N)	100
Number of generations	5000
Maximum archive size	500
Inertia weight (w)	$1/(2 \cdot \log 2)$
Cognitive/Social parameter (c)	$0.5 + \log 2$

3 Experimental Result

3.1 Configurations

The parameters used in the experiment are given in Table 1. As benchmark functions, seven DTLZ functions were employed [8]. The number of objectives was set to three for every DTLZ function. The number of variables of each DTLZ function was set to 11 for DTLZ1, 16 for DTLZ2 - DTLZ6, and 26 for DTLZ7 function.

Two performance metrics, hypervolume, i.e. the size of dominated space, and diversity measure, were employed to evaluate the performance of the MOPSO-HD. Brief explanation of the metrics is provided in the following. The size of dominated space, $\mathcal{S}$ is defined by the hypervolume of nondominated solutions [9]. The reference point to calculate $\mathcal{S}$ was set to (10, 10, 10). The quality of obtained solution set is high if this space is large. Diversity, $\mathcal{D}$ is to evaluate the spread of nondominated solutions, which is defined as follows [10]:

$$\mathcal{D} = \frac{\sum_{k=1}^{n}(f_k^{(max)} - f_k^{(min)})}{\sqrt{\frac{1}{|N_0|}\sum_{i=1}^{|N_0|}(d_i - \bar{d})^2}} \tag{2}$$

where N_0 is the set of nondominated solutions, d_i is the minimal distance between the ith solution and the nearest neighbor, and $\bar{d}$ is the mean value of all d_i. $f_k^{(max)}$ and $f_k^{(min)}$ represent the maximum and minimum objective function values of the k-th objective, respectively. A larger value means a better diversity of the nondominated solutions.

3.2 Comparison Result

Since multiple processors shared the overall optimization task, MOPSO-HDC could obtain the optimized solutions more quickly. Table 2 shows the average evaluation time ratio over 50 runs. As the table shows, the evaluation time decreases with increasing number of processors for every DTLZ function. Since every processor is synchronized at each iteration, time loss can be occur. That is why the ratio is not exactly the same with the inverse of the number of processors.

Table 2: Evaluation time ratio

Problem	$\frac{t_{\text{two processors}}}{t_{\text{one processor}}}$	$\frac{t_{\text{three processors}}}{t_{\text{one processor}}}$	$\frac{t_{\text{four processors}}}{t_{\text{one processor}}}$
DTLZ1	0.38	0.32	0.30
DTLZ2	0.51	0.40	0.31
DTLZ3	0.83	0.76	0.60
DTLZ4	0.47	0.38	0.30
DTLZ5	0.50	0.42	0.32
DTLZ6	0.44	0.39	0.30
DTLZ7	0.47	0.40	0.31

Table 3: Hypervolume and diversity measure [average(standard deviation)]

Problem	Hypervolume		Diversity	
	One processor	Four Processors	One processor	Four Processors
DTLZ1	884.37(51.51)	998.20(9.28)	29.18(17.28)	86.47(35.55)
DTLZ2	999.19(0.03)	998.74(0.39)	78.32(2.77)	78.24(3.10)
DTLZ3	100.04(221.28)	399.15(380.81)	16.48(10.08)	33.19(22.18)
DTLZ4	999.31(0.02)	998.35(1.63)	58.10(0.86)	57.92(0.96)
DTLZ5	988.64(0.23)	987.46(1.89)	54.54(0.18)	54.54(0.19)
DTLZ6	989.01(0.04)	985.96(2.59)	54.09(0.01)	54.17(0.71)
DTLZ7	708.07(1.53)	707.45(1.67)	59.07(0.26)	59.21(0.27)

Table 4: Hypothesis test (H_0: $X_{\text{four processors}} - X_{\text{one processor}} > 0$, Significance level: 0.05)

Problem	Hypervolume			Diversity		
	p-value	Reject H_0	Bayes Factor	p-value	Reject H_0	Bayes Factor
DTLZ1	0.6261	No	4.2747	0.9999	No	335.0568
DTLZ2	0.4991	No	5.0990	0.4979	No	5.0989
DTLZ3	0.9996	No	57.2779	0.9954	No	5.5556
DTLZ4	0.4981	No	5.0989	0.4938	No	5.0984
DTLZ5	0.4977	No	5.0989	0.5005	No	5.0990
DTLZ6	0.4939	No	5.0984	0.5031	No	5.0989
DTLZ7	0.4982	No	5.0990	0.5046	No	5.0987

The hypervolume and diversity of MOPSO-HDC are shown respectively in Table 3. The values were averaged over 50 runs. As the table shows, both hypervolume and diversity of MOPSO-HDC with four processor were similar to those of MOPSO with single processor. This means that there is no significant difference between the solution qualities of the two cases. To confirm this result clearly, hypothesis test was performed with those 50 pairs of sample data. Null hypothesis for every metric was defined as $X_{\text{four processors}} - X_{\text{one processor}} > 0$ and significance level was set to 0.05. As shown in Table 4, for every DTLZ function and metric, the null hypothesis could not be rejected. Bayes factor was also sufficiently large for every DTLZ function and metric which means that the odds are the null hypothesis will be turned out to be

correct. As a result, the evaluation time was significantly reduced by the proposed framework without any loss of overall solution quality and diversity.

4 Conclusion

This paper proposed a homogeneous distributed computing (HDC) framework for multiobjective evolutionary algorithm (MOEA). As an implementation of this framework, multi-objective particle swarm optimization with homogeneous distributed computing (MOPSO-HDC) was also introduced. It could solve the multi-objective optimization problems (MOPs) distributing its work to several processors homogeneously. As a result, the algorithm became not only faster but also fault-tolerant and independent to the number of processors. The effectiveness of this algorithm was demonstrated by empirical comparison between the results with the different numbers of processors, one and four. The comparison result indicated that the evaluation time was significantly reduced by the proposed framework without any loss of overall solution quality and diversity.

Acknowledgements This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012-0000150).

References

1. Kim Y-H, Kim J-H, and Han K-H (2006) Quantum-inspired multiobjective evolutionary algorithm for multiobjective 0/1 knapsack problems. Paper presented at IEEE Congress on Evolutionary Computation, pp. 2601–2606.
2. Lee K-B and Kim J-H (2008) Mass-spring-damper motion dynamics-based particle swarm optimization. Paper presented at IEEE Congress on Evolutionary Computation, pp. 2348–2353.
3. Lee K-B and Kim J-H (2009) Particle swarm optimization driven by evolving elite group. Paper presented at IEEE Congress on Evolutionary Computation, pp. 2114–2119.
4. Lee K-B and Kim J-H (2011) Multi-Objective Particle Swarm Optimization with Preference-based Sorting. Paper presented at IEEE Congress on Evolutionary computation.
5. Deb K, Zope P, and Jain A (2003) Distributed computing of pareto-optimal solutions with evolutionary algorithms. Evolutionary Multi-criterion optimization.
6. Tan K-C ,Yang Y-J, and Goh C-K (2006) A distributed cooperative coevolutionary algorithm for multiobjective optimization. IEEE Transactions on Evolutionary Computation 10(5):527–549.
7. Coello C and Lechuga M (2002) MOPSO: A proposal for multiple objective particle swarm optimization. Paper presented at IEEE Congress on Evolutionary Computation, pp. 1051–1056.
8. Deb K, Thiele L, Laumanns M, and Zitzler E (2002) Scalable multi-objective optimization test problems. Paper presented at IEEE Congress on Evolutionary Computation, pp. 825–830.
9. Zitzler E (1999) Evolutionary algorithms for multiobjective optimization: Methods and applications. Doctoral dissertation ETH 13398, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.

10. Li H, Zhang Q, Tsang E, and Ford J (2004) Hybrid estimation of distribution algorithm for multiobjective knapsack problem. Evolutionary Computation in Combinatorial Optimization, pp. 145–154.
11. Rivera W (2001) Scalable parallel genetic algorithms. Artifitial Intelligence Review 16(2):153–168.
12. Raquel C and Naval Jr P (2005) An effective use of crowding distance in multiobjective particle swarm optimization. Paper presented at Conference on Genetic and evolutionary computation, pp. 257–264.