ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 9, No. 2, 2014, *pp*.097-105

A Survey on Parallel Evolutionary Computing and Introduce Four General Frameworks to Parallelize All EC Algorithms and Create New Operation for Migration

Amin Majd [1] and Golnaz Sahebi [2]

[1] Aras International Campus, University of Tabriz, AminMajd89@Ms.Tabrizu.Ac.Ir

[2] Aras International Campus, University of Tabriz, Sahebi-Golnaz90@Ms.Tabrizu.Ac.ir

(*Received May30, 2013, accepted November09, 2013*)

Abstract. Optimization and solving NP-hard problems are very important and Evolutionary Computing (EC) methods are useful and popular. There are different types of EC methods that most of them are sequential and some others have parallel implementation. In first step we want to review some parallel implementation of EC methods and in second step we introduce four general frameworks to parallelize all EC algorithms that they are Master-Slave method, Hybrid method, Simple Multi-population method and Repulsive Multi-Population method. Finally we create a new operation for migration to keep population diversity.

Keywords: parallel genetic algorithms, parallel PSO, parallel ant colony optimization, parallel bee colony optimization, parallel memetic algorithm, repulsive operation.

1. Introduction

Optimization algorithms can be classified in heuristic and Meta heuristic. In the class of heuristic algorithms, there are construction and improvement algorithms. Meta heuristic algorithms may manage a chain or flow of executions of classical heuristics, e.g. Tabu Search, Simulated Annealing, Genetic and Memetic Algorithms.

Computability is a big problem for researchers and some problems such as NP-Hard problems have not a suitable solution that finds the best answer in limited time. There are different methods to solve them but EC are the best and more popular than them. There are different types of EC that are useful for different problems, for example Genetic algorithms are an old method for discrete problems and utilize some regular behavior in humans body and some characteristics of them and help researchers that optimize their solutions. The other one is PSO that mimics behavior of birds when migrate to other place.

EC methods are successful to solve different problems but there are some weak points that are the important reasons of bad results. For example in some problems that have a big search space is impossible that algorithms converge to local optimum results and we can improve results only with increasing of initial population. The other problem is the speed of algorithms and sometimes answers are found after a long time. Parallel algorithms can help us to improved quality and times of results.

In the past years researchers utilized some parallel EC methods to obtain good results, e.g. Parallel Genetic Algorithms [3], Parallel PSO [6], Parallel ABC (PABC) [5], Parallel Ant Colony Optimization (PACO)[4] and Parallel Memetic[7] that are more popular than other methods. Each one implemented with different technique and different equipment and hardware. For example Parallel Genetic Algorithms implement in four categories [3]: Master-Slave Genetic Algorithms, Corse Grain Genetic Algorithms (Multi-Populations Genetic Algorithms), Fine Grain Genetic Algorithms and Hybrid Genetic Algorithms that we will discuss about them in the next section.

There are different EC methods but some of them have not parallel implementation, for example ICA is a new and efficient method that is useful for continuous problems. In this Article we want to introduce four general frameworks to parallelize all EC algorithms. They divide on four categories: the first method is Master-Slave and try to decrease run time of algorithm, the second method is Hybrid method that is more complex and faster than other categories. The third method is Simple Multi-Population that is more popular and easier than others and fourth method is Repulsive Multi-Population.

2. Pervious and Related Works

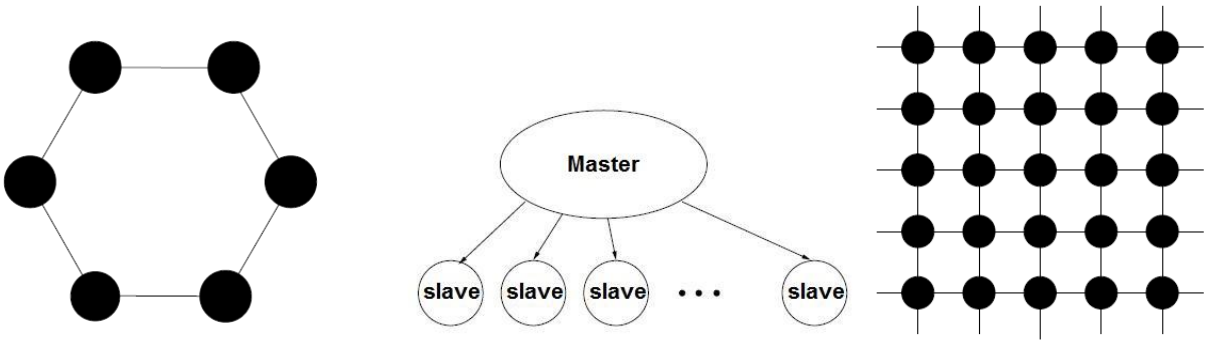
In this section we will review some parallel EC methods and try to find some common characteristic between them and utilize some of them in our implementation.

2.1. Parallel Genetic Algorithms

Genetic algorithms are population base methods that use some regular behavior of human body; each GA has an initial population and does some operations frequently that are selection, crossover, mutation and replacement. All operations are repeated until give a good result or end certain generation. A GA finds a good result when have a good selection pressure, so multi-population methods are useful because we could have bigger memory with using several processors and their memories. In multi-population methods there are several processors an each one have independent populations and each processor runs a simple GA, after certain generations all processor will stop and send some chromosome (Migration) with certain strategy, e.g. best or worse, and share results of solutions together. In this method there are some important parameters, migration rate, migration gap, topologies. Master-Slave method is the other methods.

In Master-Salve one processor is master and does all important operations of GAs such as crossover, mutation, replacement and selection and the other processor that called slave only evaluates fitness function and send back results to master processor. These methods can be implemented as synchronous and asynchronous. Synchronous methods behavior is same as simple GAs but with better runtime.

Fine grain are suitable method for parallel computing with massive processors and each processor can communicates to neighborhood processors and each individual can recombines with each individual on neighborhood processors. Speed of this methods are good but is not economical.



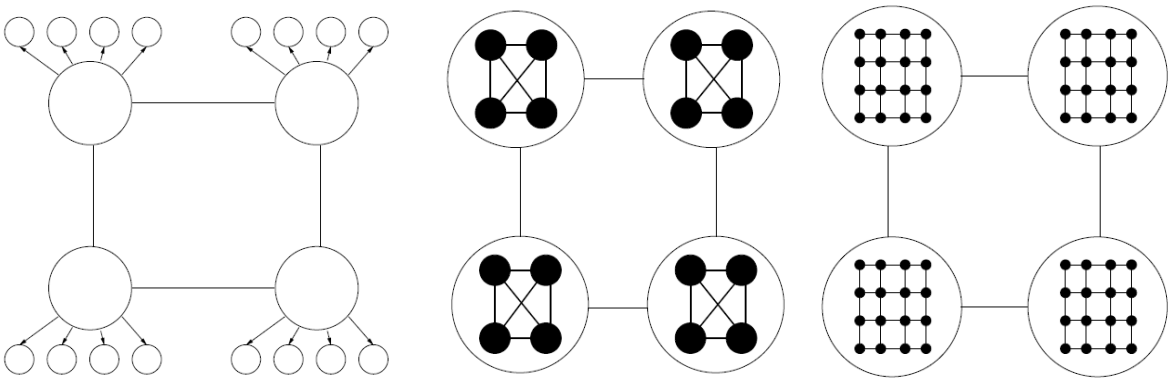
(a) (b) (c)

Fig 1: (a) Multi-population schema (b) Master-Slave schema (c) Fine grain schema [3].

Hybrid GAs are compound methods and create from two levels, upper level uses multi-population method and in lower level utilize each one of multi-population or master-slave or fine grain. This method is more efficient and faster than other methods because in this method we utilize capability of each method alone.

2.2. Parallel Ant Colony Optimization

Now we want to have a review on parallel ACO categories. ACO is an evolutionary algorithm that mimics behavior of ant to find destination. Ants use pheromone to select shortest way to find food. There are two important methods to implement parallel ACO, the first one is PACO and other one is PACO-CGD.

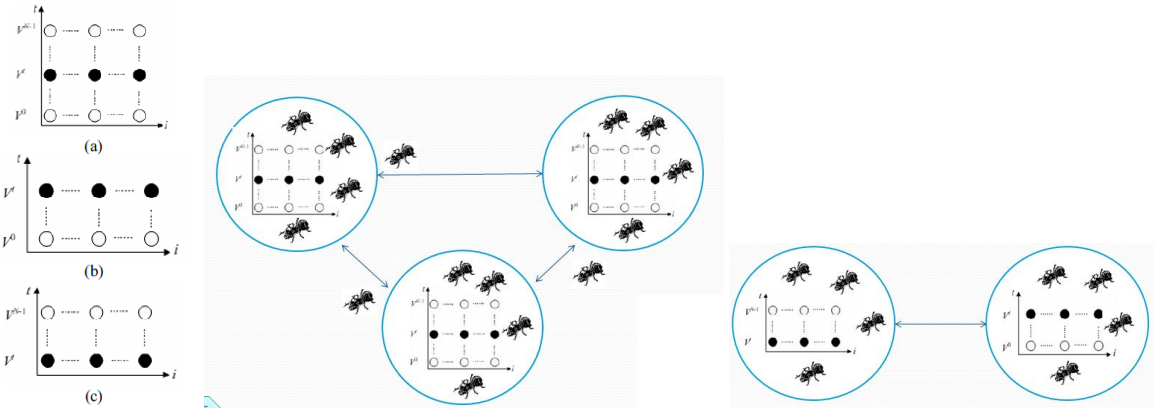


(a) (b) (b)

Fig 2: (a) Hybrid schema (high level is multi-population and low level is master-slave) (b) Hybrid schema (high level and low level are multi-population) (c) Hybrid schema (high level is multi-population and low level is fine grain) [3].

PACO method is a multi-population method that each processor has an independent population, each processor runs sequential ACO independently and after a certain frequency sends some useful information to other processors and the name of this operation is "migration". There are some important parameters like migration gap, migration rat, migration selection strategy and migration replacement strategy. Migration gap is numbers of iterations between two migrations and migration rate is number of elements that should send in each migration. Migration selection strategy divides to two categories: the best and randomize, migration replacement divides to three categories: the best, the worst and randomize that are the same parallel genetic algorithm. This method is useful and helps to increase selecting pressure and find the best results faster than the serial algorithms but there are some weak points and researchers can eliminate them.

In PACO-CGD constructor graph decomposes on smaller part and each part send to each processors and each one can run ACO method on each part and the speed of this method is better than PACO method.



(a) (b) (c)

Fig 3: (a) Graph structure, (b) PACO schema, (c) PACO-CGD schema[4].

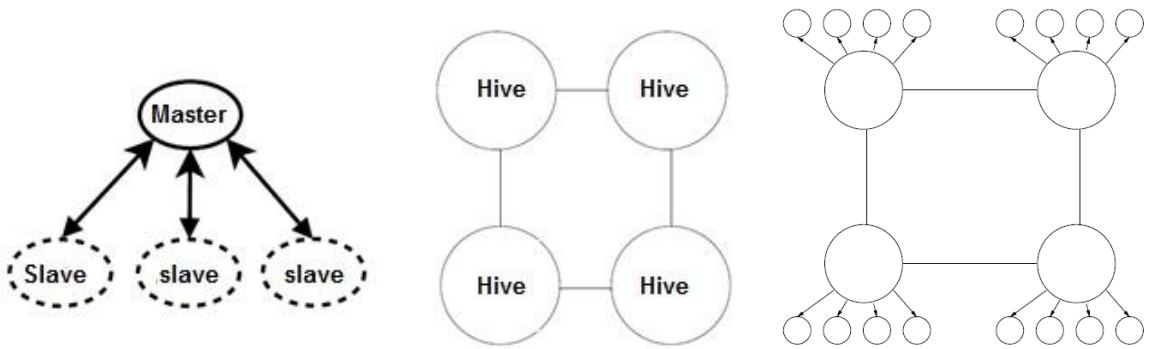
2.3. Parallel Artificial Bee Colony Optimization

Bee Colony Optimization is the other EC method that simulates behavior of bees. In this method we have three kinds of bees: scout bees, employ bees and onlooker bees. Bees show the place of food and quality of them to other bees with waggle dance. Parallel implementation of BEE has three categories for parallel BEE: master-slave, multi-population and hybrid method.

Master-slave method is the same as master-slave GAs, one processor is master and runs important operations and the other processors only evaluate individuals. This method can implements synchronous and asynchronous. Synchronous implementation has same behavior with serial BCO and asynchronous implementations have different behavior.

Multi-population method is the same as other course grain methods and each processor has an independence population and runs sequential ABC method on their population. Migration operation is available and important parameters are migration gap, migration rate and network topology.

Hybrid method is a mix method that in high level uses multi-population method and in low level uses master-slave methods and works same hybrid GAs. Hybrid method is the best and the fastest method with the most complex implementation.



(a) (b) (c)

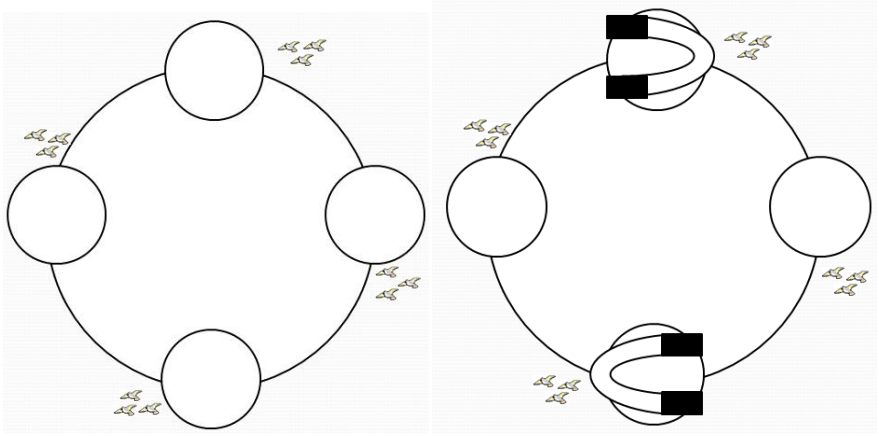
Fig 4: (a) Master-Slave schema, (b) Multi-population schema, (c)Hybrid method schema [5].

2.4. Parallel Particle Swarm Optimization

PSO is a successful method on continuous problems that parallel implementation of it is very useful and has implemented on two methods, MPSO and MRPSO.

MPSO is a multi-population method that works like other multi-population methods and each processor has different population and runs simple PSO on its population and migration operation is available. MPSO was a successful method that obtains good results on different problems.

MRPSO is an improved method of MPSO and adds an extra component of MPSO that called repulsive. Repulsive component in each processor tries to give good population diversity. The particles that migrate in the even swarms should be as different as possible to the particles already contained in these swarms, High degree of diversity in EC methods is very useful and helps me to contain a good result.



(a) (b)

Figure 5. (a) MPSO method schema (b) MRPSO method schema

2.5. Parallel Memetic Algorithm

Memetic Algorithms are population-based heuristic search approaches for optimization problems similar to genetic algorithms (GAs). GAs, however, rely on the concept of biological evolution, but MAs, in contrast and mimic cultural evolution. Parallel Memetic algorithms implemented in coarse grain method that called PARME.

PARME is a multi-population method that uses a dependence population in each processor and each processor runs Memetic algorithm dependently. There is a big difference between this method and other methods like parallel genetic algorithms. One of processors that called master, controls behavior of other processors and creates an operation table in each iteration and sends it to all processors. This operation table has values of important parameters on Memetic algorithms. The table will change in each iteration.

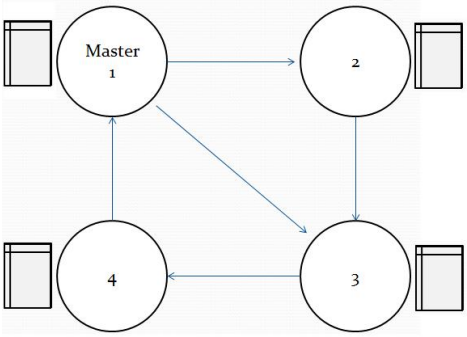


Fig 6: PARME method schema

3. Four General Frameworks to Parallelize All EC Algorithms

In this paper we review five parallel EC methods and describe all important techniques of them, we can find some common techniques that have been repeated on all parallel methods.

Multi-population methods are one of the common techniques that are more popular than others. Master-Slave methods are other methods that used in different parallel implementation. Hybrid methods are the other methods that are common on more parallel implementation that are faster than others.

We want to introduce four General frame works to parallelize all EC methods that we categories them on Multi-Population methods, Master-Slave methods and Hybrid methods that describe them on next sections.

TABLE I. TABLE OF OPERATIONS AND SYMBOLS

Title of operation	Symbols
Run serial EC methods dependency	
Independent initial population	
Run selection, exploration, exploitation and replacement operation	
Manage algorithm	
Evaluate fitness function	
Migration	
Repulsive operation	

3.1. Master-Slave Method

Master-Slave method is a method to solve the problems that their fitness function evaluation is complex. The cost of fitness function evaluation is different in different problems and can creates some delay to solving problems with serial methods, we can divide them to several groups and send each one to a processor and all processors evaluate them in the same time and send results to a main processor. This way is like the master-slave methods.

In master-slave methods we have some different processors and we divide all population to different groups and send each one to a processor and each one evaluate fitness function of each elements. In this method a processor is master and does all important operation of EC method and manages parallel methods. Master processor divides all elements to some different groups and sends them to other processors and each one evaluates fitness function and sends back results to master processor. These processors called slave processor and only evaluate fitness function of their group.

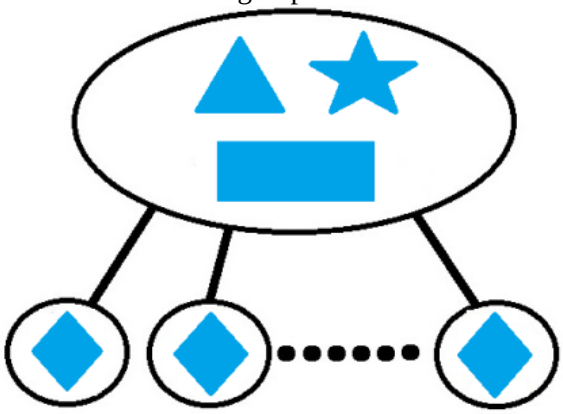


Fig 7: General framework of Master-Slave method

Master processor does all important operations of EC methods like create population, selection, exploitation, exploration and replacement. This method can be implemented as synchronous and asynchronous. Behavior of synchronous method is like serial methods but asynchronous method has different behavior of serial methods.

Master–slave method is success when we use share memory and each processors can use memory independently. In figure 7 we illustrate general framework of repulsive multi-population schema

3.2. Hybrid Method

Hybrid method is a compound method that creates with combining of two methods. We select two methods that use all benefits of the other methods to contain better results. In our hybrid method we use two methods in two level, high level and low level. In high level we use multi-population method that use its benefit (high selection pressure) and in low level we use master-slave method that increase speed of algorithms and help to evaluate fitness function faster than multi-population method. Implementation of this method is more complex than each other singly but we can obtain better speed and efficiency.

In this methods each processor in high level are master processors and other processors in low level are slave. Each master processor runs all operation of serial EC method and slaves' processors only evaluate fitness function. After a certain repetition all master processors halt their action and send some of its elements to other master processor (migration). Migration has different policy to selection and replacement. Selection policy is categorizes to two way, the best and random and replacement policy are categorize to three way, the best, the worse and random. In the best policy we select the best elements to selection or replacement and in the worse policy we select the worse elements to selection or replacement and in random policy we select some elements randomly. In figure 8 we illustrate general framework of hybrid schema.

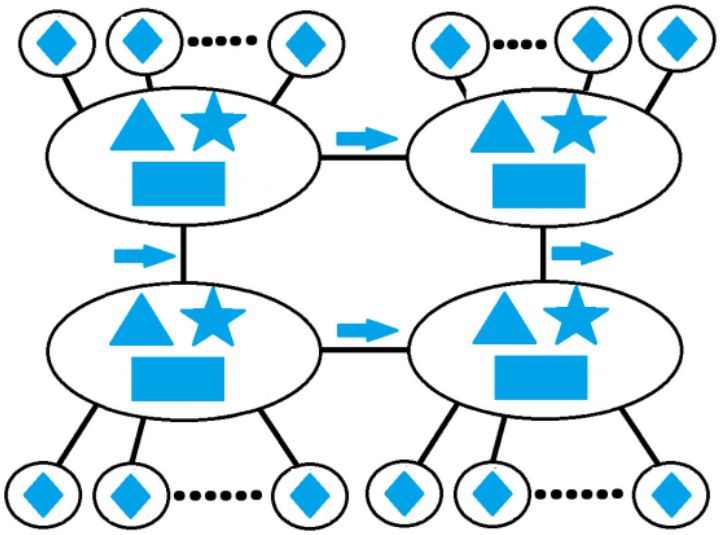


Fig 8: General framework of Hybrid method

3.3. Simple Multi-Population

All EC methods are population base and selecting pressure is very important to obtain a good solution. Memory and speed of processor are limit and we can select limited numbers of initial population and run EC algorithms on them. This problem may converge to best local optimum solutions. We can solve this problem by using Multi-Population methods to increase number of initial population and increase selecting pressure. It is a logical technique that can increase probability finding the best solution.

In multi-population method we have two or more processors and we should create dependence population in each processor and run serial EC method on each processor. In this method after certain iteration each processor sends some important elements to other processor and they should replace on some elements on destination processor. This operation is migration and there are different policy to select elements to migration and different policy to replacement element on destination processor. Selecting policy are the best or random, in the best policy processor selects best elements to migration and sends them to other processor and in random policy processor select some elements randomly. Replacement policy divided to three categories, the best, the worse and random. In the best method destination processors select the best elements to replacement, in the worse select the worse and in random method select some elements randomly to replacement. We should select some different important parameters like connection topology, migration rate and migration gap. Migration rate is the number of elements that transfer to other processors and migration gap is the distance between migrations. Connection topology is related to migration rate and migration gap for example if we want to have short migration gap with high degree of migration rate we should use full connection

topology and if we want to have spars connection topology we should use an algorithms with long migration gap and low degree of migration rate. In figure 9 we illustrate general framework of multi-population schema

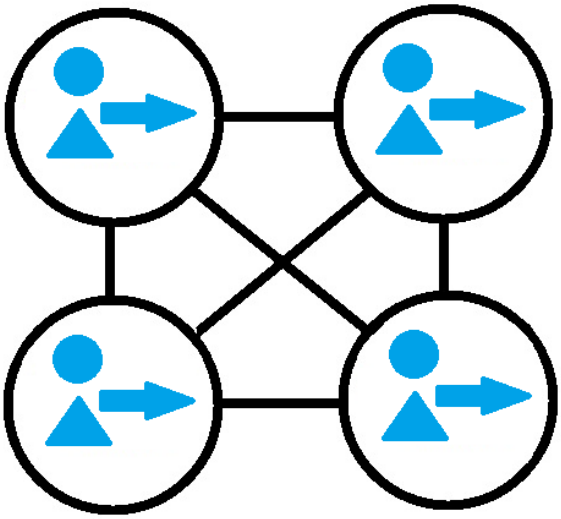


Fig 9: General framework of Multi-population method

3.4. Repulsive Multi-Population Method

All EC methods are population base and selecting pressure is very important to obtain a good solution. Memory and speed of processor are limit and we can select limited numbers of initial population and run EC algorithms on them. This problem may converge to the best local optimum solutions. We can solve this problem with using Multi-Population methods to increase number of initial population and increase selecting pressure. It is a logical technique that can increase probability finding the best solution. But sometimes migration causes that algorithm converge to a local optimum result. For improving of this problem we can use a new operation. Repulsive operation is a new operation that uses to create diversity in all population of all processors.

In repulsive multi-population method we have two or more processors and we should create dependence population in each processor and run serial EC method on each processor. In this method after certain iteration each processor sends some important elements to other processor and they should replace on some elements on destination processor. This operation is migration and there are different policy to select elements to migration and different policy to replacement element on destination processor. Selecting policy are the best or random, in the best policy processor selects best elements to migration and sends them to other processor and in random policy processor select some elements randomly. Replacement policy divided to three categories, the best, the worse and random. In the best method destination processors select the best elements to replacement, in the worse select the worse and in random method select some elements randomly to replacement. We should select some different important parameters like connection topology, migration rate and migration gap.

Repulsive operation runs when we send some elements to other processor, before destination processor receives these elements. Repulsive operation check modality of elements and if modality of migrated elements and our population modality be different then the operation publishes a permission for replacement and if modality of migrated elements and our population modality be the same then repulsive runs a exploration operation same mutation in GA or revolution in ICA afterward publishes a permission for replacement.

Connection topology is related to migration rate and migration gap for example if we want to have short migration gap with a high degree of migration rate we should use full connection topology and if we want to have spars connection topology we should use an algorithm with long migration gap and low degree of migration rate. In figure 10 we illustrate general framework of repulsive multi-population schema.

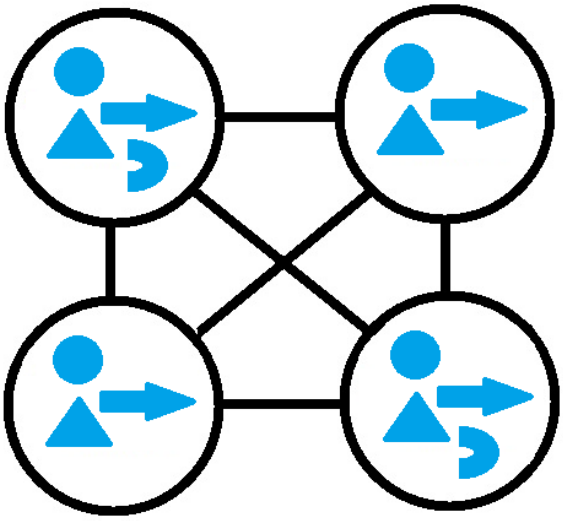


Fig 10: General framework of Repulsive Multi-Population method

4. Conclusions

In this paper we reviewed some parallel implementation of EC methods like parallel genetic algorithm, parallel ant colony optimization, parallel artificial bee colony, parallel particle swarm optimization and parallel memetic and found some command behavior of them and developed them to introduce four general frameworks to parallelize all other serial EC methods.

Each one of them has different benefit that we could choose one of them. Simple multi-population methods were useful when we had a problem with a big search space and we could increase selection pressure. Master-slave methods were useful when we had a problem with a very complex fitness function and we could decrease time of algorithm with dividing computing jabs to slave processors. Hybrid methods were useful when we had a problem with a big search space and complex fitness function and we combined multi-population and master slave and obtained very fantastic results. And when we were using repulsive multi-population we could use all benefits of simple multi-population and kept population diversity and tried to prevent of convergence to local optimum results.

In future we will implement these methods to parallelize Imperialistic Competitive Algorithms and will test them on different benchmarks and will obtain good results.

5. References

[1] H, Lotfi, A. Boroumandnia and Sh. Lotfi , *Task Graph Scheduling in Multiprocessor Systems Using a Coarse Grained Genetic Algorithm*, 2nd International Conference on Computer Technology and Development, IEEE, 2010.N.

[2] H. Lotfi, Sh. Lotfi and A. Boroumandnia , *Task Graph Scheduling in Multiprocessor Systems Using a Two Population Genetic Algorithm*, International Conference on Software and Computing Technology, IEEE, 2010.

[3] E. Cant ú-Paz, *A Survey of Parallel Genetic Algorithms*, Department of Computer Science and Illinois Genetic Algorithms Laboratory University of Illinois at Urbana-Champaign, 1997.

[4] H. Liu, P. Li and Y. Wen , *Parallel Ant Colony Optimization Algorithm*, World Congress on Intelligent Control and Automation, China, June, 2006.

[5] R. Parpinelli, C. Benitez and S. Lopes , *Parallel Approaches for the Artificial Bee Colony Algorithm*, Handbook of Swarm Intelligence, Vol. 8, pp. 329-345, 2010.

[6] L. Vanneschi, D. Codecasa and G. Mauri , *A Comparative Study of Four Parallel and Distributed PSO Methods*, New Generation Computing, Vol. 29, pp. 129-161, April 2011.

[7] J. Digalakis and K. Margaritis , *A Parallel Memetic Algorithm for Solving Optimization Problems*, 4th Metaheuristics International Conference, Parallel Distributed Processing Laboratory, Greece, 2001.

[8] H. Narasimhan, *Parallel Artificial Bee Colony (PABC) Algorithm,* World Congress on Nature & Biologically Inspired Computing, pp. 306-311, IEEE, 2009.

[9] A. Majd, Sh. Lotfi and G. Sahebi, *Review on Parallel Evolutionary Computing and Introduce Three General Framework to Parallelize All EC Algorithms,* The 5th Conference on Information and Knowledge Technology, IEEE, 2013.

[10] E. A. Gargari and C. Lucas, *Imperialist Competitive Algorithm: An Algorithm for Optimization Inspired by Imperialistic Competition,* Congress on Evolutionary Computation, IEEE, 2007.

[11] H. Narasimhan, *Parallel Artificial Bee Colony (PABC) Algorithm,* World Congress on Nature & Biologically Inspired Computing (NaBIC), IEEE, 2009.

[12] Z. Yang and B. Yu, *A Parallel Ant Colony Algorithm for Bus Network Optimization,* Computer-Aided Civil and Infrastructure Engineering, vol. 22, pp. 44–55, 2007.

[13] E. Alba, *Parallel Evolutionary Algorithms Can Achieve Super-Linear Performance,* Information Processing Letters, vol. 82, pp. 7–13, ELSEVIER, 2002.

[14] L. Vanneschi, D. Codecasa and G. Mauri, *A Comparative Study of Four Parallel and Distributed PSO Methods,* New Generation Computing, vol 29, pp. 129-161, Ohmsha Ltd. and Springer, 2011.

[15] J. Digalakis and K. Margaritis, *A Parallel Memetic Algorithm for Solving Optimization Problems,* 4th Metaheuristics International Conference, MIC, 2001.

[16] E. C. G. Wille and E. Y. H. S. Lopes, *Discrete Capacity Assignment in IP networks using Particle Swarm Optimization,* Applied Mathematics and Computation, vol 217, pp. 5338–5346, ELSEVIER, 2011.

[17] A. Mousa, W. Wahed and R. Allah, *A Hybrid Ant Colony Optimization Approach Based Local Search Scheme for Multi Objective Design Optimizations,* Electric Power Systems Research, vol 81, pp. 1014–1023, ELSEVIER, 2011.

[18] P. Y. Yin, S .S Yu, P. P. Wang, Y. T. Wang, *A Hybrid Particle Swarm Optimization Algorithm for Optimal Task Assignment in Distributed Systems,* Computer Standards & Interfaces , vol 28 , pp. 441–450, ELSEVIER, 2006.

[19] H. Bahrami, K. Faez and M. Abdechiri, *Imperialist Competitive Algorithm using Chaos Theory for Optimization (CICA),* 12th International Conference on Computer Modeling and Simulation, 2012.

[20] M. Abdechiri, K. Faez and H. Bahrami, *Adaptive Imperialist Competitive Algorithm (AICA),* 9th IEEE International Conference on Cognitive Informatics (ICCI), 2010.

[21] H. Bahrami, M. Abdechiri and M. R. Meybodi, *Imperialist Competitive Algorithm with Adaptive Colonies Movement, I.J. Intelligent Systems and Applications* , Vol 2, pp. 49-57, 2012.

[22] E. Alba, F. Luna, A. J. Nebro and J. M. Troya, *Parallel Heterogeneous Genetic Algorithms for Continuous Optimization,* Parallel Computing, vol 30, pp. 699–719, ELSEVIER, 2004.

[23] K. Weinert, J. Mehnen and G. Rudolph, *Dynamic Neighborhood Structures in Parallel Evolution Strategies,* Complex Systems Publications, vol 13, pp.227–243, 2002.

[24] Y. Zhou and Y. Tan, *Particle Swarm Optimization with Triggered Mutation and its Implementation Based on GPU,* ACM, pp. 1-8 ,2010.

[25] A. Basturk, R. Akay and A. Kalinli, *Comparison of fine-grained and coarse-grained parallel models in particle swarm optimization algorithm,* 2nd World Conference on Information Technology (WCIT)-2011.

[26] R, Dewri, and N, Chakraborti, *Simulating recrystallization through cellular automata and genetic algorithms,* Modelling Simul. Mater. Sci. Eng. 13 (2005), pp. 173-183.

[27] A. Gray, *Modern Differential Geometry,* CRE Press, 1998.