Empirical comparison of level-wise hierarchical multi-population genetic algorithm*

Tzung-Pei Hong\textsuperscript{a,b}, Yuan-Ching Peng\textsuperscript{b}, Wen-Yang Lin\textsuperscript{a} and Shyue-Liang Wang\textsuperscript{c}

\textsuperscript{a}Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan; \textsuperscript{b}Department of Computer Science and Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan; \textsuperscript{c}Department of Information Management, National University of Kaohsiung, Kaohsiung, Taiwan

\section*{ABSTRACT}

Metaheuristics have recently been commonly used to solve complex problems in real applications. Some scholars used multiple populations in metaheuristic approaches to shorten the execution time in finding nearly optimal solutions. In this paper, we revisit the properties of sub-population execution for genetic algorithm (GA) and design a level-wise hierarchical sub-population architecture. We also make experiments for comparing the performance of the architecture for unimodal and multimodal functions. The experimental results show that using the hierarchical GA architecture produces a more significant improvement on multimodal functions than on unimodal ones.

\section*{1. Introduction}

In the past decades, many metaheuristics were generated from very simple concepts of nature. The inspiration could be classified into the following three major types (Mirjalili, Mirjalili, \& Lewis, 2014). The first type is from the evolutionary concept. In this type, the optimization is done by evolving an initial group of random solutions by evolutionary algorithms. One of the most popular algorithms in this type is the genetic algorithm (GA) (Holland, 1975).

The second type of metaheuristics is based on animals’ behaviours. These algorithms mainly mimic the social behaviour of swarms of creatures in nature. The search agents implemented by computer programs thus navigate towards solutions using the simulated collective and social intelligence of creatures. Examples include Particle Swarm Optimization (PSO) (Kennedy \& Eberhart, 1995) and Ant Colony Optimization (ACO) (Dorigo, Birattari, \& Stützle, 2006), among others.

The third type of metaheuristics is from the observation of physical phenomena. This kind of algorithms randomly generates a set of agents to search and move throughout...
the search space based on some physical phenomenon. Some famous examples are Grav-itational Local Search (GLSA) (Simon, 2008) and Gravitational Search Algorithm (GSA) (Esmat, Nezamabadi-Pour, & Saryazdi, 2009).

In this paper, we focus on designing hierarchical execution process for GAs and compare its performance on functions with different numbers of modals. The first GA was proposed by Holland in 1975 (Holland, 1975). The idea came from Charles Darwin’s theory about ‘natural selection, survival of the fitness’ (https://en.wikipedia.org/wiki/Charles_Darwin). GA is a heuristic search procedure based on the mechanism of natural selection, genetics, and evolution. During these years, GA has been widely applied in many fields such as bioinformatics (Wasserman & Sandelin, 2004), optimization design (Samii & Michielssen, 1999), machine learning (Goldberg & Holland, 1988), knowledge systems (Nara, Univ, & Shiose, 1992), and manufacturing scheduling (Wall, 1996).

Multi-population genetic algorithms (MGAs) have been recognized as being efficient and effective in finding nearly optimal solutions (Cantú-Paz, 1998). In this paper, we study the characteristics and execution of GA on a hierarchical tree structure. We first propose an algorithm called hierarchical multi-population genetic algorithm (HMGA), in which, the execution process is represented as a tree structure and conducted in a level-wise manner. The given total number of chromosomes are divided into several sub-populations, which are then put at leaf nodes of a tree structure and move up along with the tree when some criteria are met. Each node executes a traditional GA algorithm with the sub-population assigned. We expect that the search diversity can increase in this way because of the sub-populations and their layer-by-layer merge, and thus the final performance may be improved.

The rest of the paper is organized as follows. Section 2 reviews some related studies, including GAs, MGAs, and hierarchical structures. The adopted HMGA is described in Section 3. Experiments on unimodal and multimodal functions are shown in Section 4, with comparison and discussion. Finally, conclusion and future work are stated in Section 5.

2. Related works

In this section, some related studies on GAs and MGAs are briefly reviewed.

2.1. Genetic algorithms (GAs)

Holland firstly introduced the GAs in 1975 (Holland, 1975). It is categorized as a global search metaheuristic that mimics the process of natural selection. The metaheuristic is often used to find true or approximate solutions to optimization and search problems (Mitchell, 1998). GAs is a particular class of evolutionary algorithms, which generate solutions using the techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover (also called recombination). In a GA, a population composed of candidate solutions is maintained, and each candidate is formed by a set of genes, as shown in Figure 1. The genes may represent the properties of the solution and can be mutated and crossovered to evolve towards better solutions (https://en.wikipedia.org/wiki/Genetic_algorithm).
Candidate solutions may be encoded in different ways. For numerical solutions, binary-coded representation and real-coded representation are two primary ways. After coding, each chromosome is represented by a string of bits or numbers. Then a GA executes the required crossover and mutation operators to make the population evolve according to the specified evaluation function.

2.2. Multi-population genetic algorithms

The multi-population genetic algorithm (MGA), which was first proposed by Grefenstette (1981), is an extension of the traditional single-population GAs. It divides a population into several isolated sub-populations in which individuals are allowed to exchange from one to another. Compared to the single-population GA, the MGA is more similar to the natural GA. MGA is able to raise the parallelism of the genetic execution and is more resistant to premature convergence than the single-population one.

Multi-population genetic algorithms (MGAs) have been recognized as being efficient and effective in finding nearly optimal solutions (Cantú-Paz, 1998). During these years, different variants based on MGA have been proposed. For example, Cochran, Horng, and Fowler (2003) adopted a MGA, which used a two-stage approach to solve multi-objective scheduling problems. Zegordi and Nia (2009) proposed the multi-society genetic algorithm (MSGA), which had three populations with different fitness functions for solving the transportation scheduling. Lin, Hong, Liu, and Lin (2012) proposed a general framework for studying MGAs with ring topology.

Sefrioui and Périaux (2000) proposed a hierarchical topology for the layout of three layers. The sub-populations in all the layers simultaneously ran with different resolutions. The method used parallel GAs based on the notion of sub-populations. Its main advantage lay in the interaction of the three layers. The solutions went up and down the layers, and the best ones keep going up until they get refined. In this paper, we adopt a hierarchical processing mechanism different to GA.

3. Hierarchical multi-population genetic algorithm (HMGA)

In this section, we describe the proposed HMGA. The main idea of the proposed HMGA is first explained. The basic strategy used in HMGA is the divide-and-conquer in which the whole execution is organized into a tree hierarchy. An original GA population is first
divided into several sub-populations. Then each sub-population is placed in a leaf node of the tree structure and is processed by the GA algorithm individually. When the termination criterion is met, the sub-populations that belong to the same parent node are merged and conceptually placed in the parent node. Then the same procedure is repeated until the root node. The best result from the root node is finally output as the solution.

Formally, let the height of the hierarchical tree structure be denoted $h$ and the branching degree denoted $b$, respectively. Also let $p$ represent the total chromosome number at the nodes of the same levels, and $S_{i,j}$ denotes the sub-population that is located at the $j$-th node (from the left) of the $i$-th level. Note that here the root is at level 1. Figure 2 illustrates the above idea with $h = 3$ and $b = 2$.

For the hierarchical MGA in Figure 2, its execution process can be shown in Figure 3. In Figure 3, the total $N$ chromosomes are first divided into four sub-populations and put into the four leaf nodes of the execution tree. Each sub-population in a leaf node is then executed by the GA. When the termination criterion for the HMGA is met, like achieving a certain number of generations, both the sub-populations belong to the same parent node at the next high level (i.e. level 2). For example, $P_{3,1}$ and $P_{3,2}$ will be merged into $P_{2,1}$, as Figure 4 shows.

With the above idea, the detailed steps of the proposed HMGA are described as follows.

**The Hierarchical Multi-population Genetic Algorithm (HMGA):**

**INPUT:**

1. A given fitness function $F$ corresponding to the problem to be solved,
2. A given GA with a crossover rate $P_c$ and a mutation rate $P_m$,
3. A fixed total number $N$ of chromosomes at each level,
4. Height $h$ and branching degree $b$ of the hierarchical execution,
5. A total generation number $T$.

**OUTPUT:** A nearly optimal solution for the fitness function $F$.

**STEP 1:** Initially set the current generation number $g$ as 0, where $g$ is used to keep the current iteration number from the beginning of the algorithm.

**STEP 2:** Randomly generate $N$ chromosomes and equally allocate them into $b^{h-1}$ sub-populations, with each has $\left\lfloor \frac{N}{b^{h-1}} \right\rfloor$ or $\left\lfloor \frac{N}{b^{h-1}} \right\rfloor + 1$ chromosomes.

**STEP 3:** Evaluate the fitness values of the chromosomes in each sub-population.

![Figure 2](image-url) An example of HMGA with $h = 3$ and $b = 2$. 
Figure 3. The execution process of the proposed HMGA for $h = 3$ and $b = 2$.

Figure 4. Merging sub-populations at a lower-level into one at a higher level.
STEP 4: Set the current level number $l$ as $h$, where $l$ is used to keep the current level number.

STEP 5: For each sub-population in Level $l$, do the following substeps.

SUBSTEP 5.1: Execute the crossover operations on the chromosomes. Here we use the roulette wheel to select chromosomes for crossover.

SUBSTEP 5.2: Execute the mutation operations on the chromosomes.

SUBSTEP 5.3: Evaluate the fitness values of the offspring chromosomes.

SUBSTEP 5.4: Execute the selection mechanism to generate the population at the next generation.

STEP 6: Set $g = g + 1$.

STEP 7: If $g = T$, then do Step 10 to finish the algorithm; If $l$ is equal to 1 go to STEP 5; If $g = (\lfloor T/l \rfloor * (h – l + 1))$, do the next step (STEP 8); Otherwise, go to STEP 5 to execute for the next iteration.

STEP 8: Merge the sub-populations with the same parent into the one at level $l–1$.

STEP 9: Set $l = l–1$ and go to STEP 5.

STEP 10: Stop the execution and output the best solution in the root population to users.

Note that different selection mechanisms can be chosen in STEP 5. For example, the elite or roulette wheel strategies can be applied here.

4. Experimental evaluation

A series of experiments were conducted to evaluate the performance of the proposed algorithms. Four benchmark functions were used to test the performance of the proposed approaches. These functions could be divided into two groups: unimodal problems and multimodal problems (Jamil & Yang, 2013). Tables 1 and 2 list the ranges of the search spaces and the best values respectively for the unimodal and the multimodal functions (Jamil & Yang, 2013).

The Sphere and Rosenbrock functions belong to the first group. These two functions are relatively simple. It is worth mentioning that the Rosenbrock function is sometimes treated as a multimodal function. Here we put it in the first group since its characteristics are more

| Table 1. The unimodal functions used in the experiments. |
|-------------------------------------------------------|
| Function (F1) | $f_1(x) = \sum_{i=1}^{d} x_i^2$ | [-5.12, 5.12] | 0 |
| Sphere (F1) | | | |
| Rosenbrock (F2) | $f_2(x) = \sum_{i=1}^{d-1} \left(100(x_{i+1} - x_i)^2 + (x_i - 1)^2\right)$ | [-2.048, 2.048] | 0 |
| | | | |
| Table 2. The multimodal functions used in the experiments. |
|-------------------------------------------------------|
| Function (F3) | $f_3(x) = 10d + \sum_{i=1}^{d} [x_i^2 - 10 \cos (2\pi x_i)]$ | [-5.12, 5.12] | 0 |
| Rastrigin (F3) | | | |
| Griewank (F4) | $f_4(x) = \frac{1}{4000} \sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} \cos \left(\frac{x_i}{\sqrt{i}}\right) + 1$ | [-600, 600] | 0 |
closely tied to a unimodal function in our experiments. The remaining two functions, the Rastrigin function and the Griewank function, are multimodal. Such functions are more difficult than unimodal because they have many local optima. The 2-D landscape maps of the two unimodal and the two multimodal functions (Jamil & Yang, 2013) are shown in Figures 5 and 6, respectively.

These functions were tested by the proposed algorithms for showing the effects of hierarchical execution. The parameter settings for all the experiments were described as follows. The dimension (variable number) $d$ of each test function was set at 10. The maximum total number $T$ of generations was set at 3000. For the other parameters of GA, the crossover rate $P_c$ was set at 0.8 and the mutation rate $P_m$ was set at 0.005. Different numbers of populations were used.

A fixed total number of 100 chromosomes were used to investigate the variation of the performance of HMGA. The following six cases of hierarchical execution with different height and branch degrees were adopted and shown in Figure 7.
As the algorithm of HMGA shows, the termination criterion at each level was for sub-populations to run for $T/h$ iterations. Figures 8–11 show the results of HMGA for the four benchmark functions.

It could be observed from the above results that using hierarchical execution was useful especially on F3 and F4. According to Figures 10 and 11 for F3 and F4, HMGA had better performance than traditional GA. This is because the former could increase the diversity from merge in the execution process.

The experimental results indicated that HMGA performed well when the function was multimodal. Figure 10 for F3 is a good example. In Figure 10, it could be found that the traditional GA got trapped in a local optimal early, and the HMGA with a large height degree and a high branch degree could more easily avoid being trapped in a local optimal. The reason was HMGA could continue improving its solutions due to the merge operations. Since MHGA processed the sub-populations at each level for the same 1000 iterations ($T/h$), it could be observed that HMGA got a better solution after 1000 and 2000 iterations, because the merge operations occurred at those times.

Figure 7. Six cases of the hierarchical execution structure.

Figure 8. The evolutionary process of HMGA for function F1.
Figure 9. The evolutionary process of HMGA for function F2.

Figure 10. The evolutionary process of HMGA for function F3.

Figure 11. The evolutionary process of HMGA for function F4.
We then show the effect of HMGA on F3 with a fixed height = 2 and different branch factors. The results are shown in Figure 12. It could be seen that more branches brought better benefits because the diversity was increased.

We then show the effect of HMGA on F3 with a fixed depth = 2 and different heights. The results are shown in Figure 13. It could be seen that larger height brought better benefits because of the mixing effect.

5. Conclusion and future work

In this paper, we have revisited the multi-population mechanisms and designed a hierarchical execution architecture of GA. We have also proposed a HMGA. In the algorithm,
an original GA population is first divided into several sub-populations, each of which is placed in a leaf node of the tree structure and is processed by the GA algorithm individually. When the termination criterion is met, the sub-populations that belong to the same parent node are merged and conceptually placed in the parent node. The algorithm can also be easily executed in parallel since the nodes are homogeneous and coded the same. Several experiments have been made to verify the performance of the proposed algorithm. From the experimental results, it can be observed that the proposed approach shows good performance when compared to the single layer execution. Besides, the effect of the proposed approach on multimodal functions is better than on unimodal ones. As future works, we will conduct more experiments on other benchmarks. We will also study the effect of adding more mechanisms such as migration in the architecture.

**Disclosure statement**

No potential conflict of interest was reported by the authors.

**Note on contributors**

*Tzung-Pei Hong* received his B.S. degree in chemical engineering from National Taiwan University in 1985, and his Ph.D. degree in computer science and information engineering from National Chiao-Tung University in 1992. He served at the Department of Computer Science in Chung-Hua Polytechnic Institute from 1992 to 1994, and at the Department of Information Management in I-Shou University from 1994 to 2001. He was in charge of the whole computerization and library planning for National University of Kaohsiung in Preparation from 1997 to 2000 and served as the first director of the library and computer centre in National University of Kaohsiung from 2000 to 2001, as the Dean of Academic Affairs from 2003 to 2006, as the Administrative Vice President from 2007 to 2008, and as the Academic Vice President in 2010. He is currently a distinguished professor at the Department of Computer Science and Information Engineering and at the Department of Electrical Engineering, National University of Kaohsiung, and a joint professor at the Department of Computer Science and Engineering, National Sun Yat-sen University, Taiwan. He got the first national flexible wage award from Ministry of Education in Taiwan. He has published more than 500 research papers in international/national journals and conferences and has planned more than 50 information systems. He is also the board member of more than 40 journals and the programme committee member of more than 500 conferences. His current research interests include knowledge engineering, data mining, soft computing, management information systems, and www applications.

*Yuan-Ching Peng* received his B.S. and M.S. degrees from the Department of Computer Science and Information Engineering in National United University, Miaoli Taiwan, and the Department of Computer Science and Information Engineering from National Sun Yat-Sen University, Kaohsiung, Taiwan, in 2008 and 2016, respectively. His research interests include data mining, evolutionary algorithms, and fuzzy theory.

*Wen-Yang Lin* is a professor at the Department of Computer Science and Information Engineering, National University of Kaohsiung. He received his Ph.D. in Computer Science and Information Engineering from National Taiwan University in 1994. From 2004 to 2007, he has chaired the Department of Computer Science and Information Engineering at National University of Kaohsiung, and served as the Director of Computer Science and Information Center from 2008 to 2010. His current research interests include data mining, data warehousing, and evolutionary computation. He is also interested in applying data mining techniques to the area of Healthcare and Medical Informatics. He has co-edited several special issues of renowned international journals, (co-)authored more than 180 refereed publications, served as co-chair of programme committee, and organized special sessions for
many international conferences, including ASONAM, IEEE SMC, WCCI, and IEA/AIE. He is a member of IEEE, the Taiwanese AI Association, and Taiwanese Association for Social Networks.

Leon Shyue-Liang Wang received his Ph.D. from State University of New York at Stony Brook in 1984. From 1984 to 1994, he joined the University of New Haven and New York Institute of Technology as assistant/associate professor. From 1994 to 2002, he joined I-Shou University in Taiwan and served as director of Computing Center, chairman of Information Management Department, and director of Library. From 2003 to 2007, he rejoined NYIT. From 2009 to 2016, he was professor, chairman, Dean of College of Management, and vice president at National University of Kaohsiung, Taiwan. From August 2016, he started to serve as President of the University. He is recipient of the 2011–2014, 2014–2017 national flexible wage awards from Ministry of Education in Taiwan, and a Fellow of Institute of Engineering and Technology in UK. He is president of Taiwanese Association of Social Networks and Editor-In-Chief of International Journal of Information Privacy, Security and Integrity. He has published over 220 papers in the areas of data mining, privacy preservation, soft computing, and served PC member and session chair of over 140 international conferences.

References

Cantú-Paz, E. (1998). A survey of parallel genetic algorithms. *Calculateurs Paralleles*, 10(2), 141–171.

Cochran, J. K., Horng, S. M., & Fowler, J. W. (2003). A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. *Computers & Operations Research*, 30(7), 1087–1102.

Dorigo, M., Birattari, M., & Stützle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4), 28–39.

Esmat, R., Nezamabadi-Pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Information Sciences*, 179(13), 2232–2248.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning*, 3(2), 95–99.

Grefenstette, J. J. (1981). *Parallel adaptive algorithms for function optimization* (Technical report, pp. 18–81). Nashville, TN: Vanderbilt University.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Cambridge, MA: MIT Press.

Jamil, M., & Yang, X. S. (2013). A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2), 150–194.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *The 1995 IEEE International Conference on Neural Networks*, 4, 1942–1948.

Lin, W. Y., Hong, T. P., Liu, S. M., & Lin, J. H. (2012). Revisiting the design of adaptive migration schemes for multipopulation genetic algorithms. In Y. H. Kuo, V. S. M. Tseng, H. Y. Kao, T. P. Hong, & M. F. Horng (Eds.), *The 2012 conference on technologies and applications of artificial intelligence* (pp. 338–343). Los Alamitos, CA: IEEE.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, 69, 46–61.

Mitchell, M. (1998). *An introduction to genetic algorithms*. Cambridge, MA: MIT press.

Nara, K., Univ, H., & Shiose, A. (1992). Implementation of genetic algorithm for distribution systems loss minimum re-configuration. *IEEE Transactions on Power Systems*, 7(3), 1044–1051.

Samii, Y. R., & Michielssen, E. (1999). Electromagnetic optimization by genetic algorithms. *Microwave Journal*, 42(11), 232–232.

Sefrioui, M., & Périaux, J. (2000). A hierarchical genetic algorithm using multiple models for optimization. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, & H.-P. Schwefel (Eds.), *The sixth international conference on parallel problem solving from nature* (pp. 879–888). Berlin: Springer-Verlag.

Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation*, 12(6), 702–713.
Wall, M. B. (1996). A genetic algorithm for resource-constrained scheduling (PhD dissertation). Massachusetts Institute of Technology.

Wasserman, W. W., & Sandelin, A. (2004). Applied bioinformatics for the identification of regulatory elements. *Nature Reviews Genetics, 5*(4), 276–287.

Zegordi, S. H., & Nia, M. A. B. (2009). A multi-population genetic algorithm for transportation scheduling. *Transportation Research Part E: Logistics and Transportation Review, 45*(6), 946–959.