Search Economics: A Solution Space and Computing Resource Aware Search Method

Chun-Wei Tsai
Department of Computer Science and Information Engineering
National Ilan University
Yilan 26047, Taiwan, R.O.C.
cwtsai0807@gmail.com

***Abstract*—Although most metaheuristic algorithms claimed that they have a chance to find the optimal solution if given sufficient computation time. In fact, a metaheuristic algorithm may search the same region or particular solutions for a long time when the search process is approaching the convergence state. The question that arises now is, how to invest the limited computing resource to search for the "solutions on the right region" instead of wasting time to search for the irrelevant solutions. This paper introduces a new metaheuristic algorithm, called search economics (SE), for solving optimization problems. The basic idea of the SE is to depict the solution space based on the solutions that have been checked by the search algorithm and use the "information of solution space" to search for the solution on the convergence process. Based on these concepts, the investment of a search process will be more meaningful and thus not easy to fall into local optimum at the early iterations. The experimental results show that the proposed algorithm can provide a result that is significantly better than those provided by state-of-the-art metaheuristic algorithms in terms of the quality.**

***Index Terms*—Metaheuristic algorithms, solution space, and economics.**

I. Introduction

The successful results of metaheuristics can be easily found almost everywhere today [1]. It means that the typical application field of metaheuristics is for the situation where there is a complex optimization problem to be solved but there does not exist a possible way to find the optimal solution in a reasonable time by using limited computing resources. Unlike the exhaustive and greedy search algorithms, metaheuristic algorithm works by guessing the "right" directions for finding the possible solution; thus, it is not only faster than the exhaustive search in finding an approximate solution, but it can also find a better solution than the greedy search does. For the time consumed, the basic idea of metaheuristics is not to check all the candidate solutions so as to find the solution because for some complex optimization problems, checking every possible solution to find the solution in a reasonable time using the computers today is simply a mission impossible. Rather, it tries to use a smart method to guess where good solutions can be located in the search space. For the quality of the end result, the basic idea of metaheuristics is to embed in their search operators the mechanism to escape the local optimum or to avoid the search process from falling into local optimum at the early iterations.

Since the development of metaheuristic algorithm can be dated back to the 1950s or even earlier, the year 2,000 can be imagined as a simple watershed of these search algorithms. The discussion of a variety of developmental trajectories for the state-of-the-art metaheuristics that were presented before the year 2,000 can be found in [1], [2]. These algorithms can be divided into two groups by a well-known method, namely, single-solution-based algorithm (SSBA) versus population-based algorithm (PBA). The main difference is in the number of candidate solutions (directions) that is used at the same time in the search process. The simulated annealing [3] and tabu search [4] fall into the group of SSBA while the genetic algorithm [5], ant colony optimization [6], particle swarm optimization [7], and differential evolution [8] are in the group of PBA. After the year 2,000, still many new metaheuristics were presented for solving the complex optimization problem. The swarm intelligence has become an important research field because several new metaheuristics are inspired by the behavior of swarm [9], [10], [11]. In addition to the behavior of swarm, natural phenomena is another way of thinking in the development of new metaheuristics, such as harmony search [12], gravitational search algorithm [13], and coral reefs optimization [14].

```
1  Metaheuristic Algorithm()
2  {
3      s = Initialization()                          I
4      While the termination criterion is not met
5          v = Transition(s)                         T
6          f = Evaluation(v)                         E
7          s = Determination(v, f)                   D
8      End
9      Output s
10 }
```

Fig. 1. Outline of a metaheuristic algorithm [15].

As shown in Fig. 1, most metaheuristics contain four important operators: initialization (**I**), transition (**T**), evaluation (**E**), and determination (**D**), where s is a solution or a set of solutions, v is the state of s after application of the transition operator, and f is the fitness of v. Except the initialization operator, transition, evaluation, and determination are the main operators for each iteration. Their roles are to transit the old solution to a new one first, then evaluate the quality of the new solution, and guess the search direction by using information of the solutions searched in the previous iterations, respectively.

978-1-4799-8697-2/15 $31.00 © 2015 IEEE
DOI 10.1109/SMC.2015.447

Most metaheuristics are able to find an approximate solution very quickly compared with the exhaustive algorithm for complex optimization problems. Some redundant searches on the search process may not prevent a metaheuristic algorithm from finding the approximate solution. However, these redundant searches might bring up undesired problems. For example, searching the same solutions for a long time will make the metaheuristic algorithm have a small chance to exit the local optimum, thus cannot further improve the search results. To make each search as meaningful as possible, this study presents a novel metaheuristics algorithm by using information of the solution space, computing resource, and current solution to enhance the *value of each search*. The main contributions of this paper can be summarized as follows:

1) The main search process is designed from scratch for distributed computing; thus, no global information needs to be kept. By using the information of computing resource and solution space, the search algorithm can then determine which direction or region is worth being searched later, just like we have to understand the market circumstances before we invest. To achieve this goal, a new search algorithm, called vision search (VS), is presented in Section II-D.
2) The proposed algorithm will attempt to survey the solution space on the search process to provide the search algorithm information about the solution space that it has already checked. To achieve this goal, a new mechanism for metaheuristics, called marketing research (MR), is discussed in Section II-E.

The remainder of the paper is organized as follows. The basic idea of the proposed algorithm (SE) is given in Section II. Section III begins with a description of the simulation environment. Then, the simulation results are given, the performance of SE and the other algorithms evaluated in this paper is evaluated. Finally, Section IV draws the conclusion and gives some future research directions of this research.

II. Search Economics

A. Notations

To simplify the discussion that follows, the following notations are used throughout the rest of the paper.

- s, s_i set of investments (solutions) for the optimization problem in question; i.e., $s = \{s_1, s_2, \ldots, s_n\}$, where s_i is the investment of the i-th searcher and n the number of searchers and investments.
- r, r_j set of regions in the market (solution space); i.e., $r = \{r_1, r_2, \ldots, r_h\}$, where r_j is the j-th region of the market and h the number of regions.
- r_j^b best good (solution) of region r_j.
- m, m_{jk} set of goods at all regions in the market; i.e., $m = \{m_{11}, \ldots, m_{1w}, \ldots, m_{h1}, \ldots, m_{hw}\}$, where m_{jk} is the k-th good at the j-th region and w the number of goods at each region.
- v^i, v_{jk}^i set of possible investments of the i-th searcher, which is defined as the crossover of s_i and all goods in m. That is, $v^i = \{v_{jk}^i\}$ where $v_{jk}^i = s_i \otimes m_{jk}$ with $\otimes$ being the crossover operator.
- e, e_{ij} set of expected values; i.e., e_{ij} is the expected value for the i-th investment at the j-th region.
- t_j^a number of times the j-th region has been invested (searched). Initially, $t_j^a = 1$; its value will be increased by 1 every time the j-th region is searched.
- t_j^b number of times the j-th region has not been invested (searched). Initially, $t_j^b = 1$; its value will be increased by 1 every iteration the j-th region is not searched but will be reset to 1 if it is searched.

Note that in this study, the solution space is called the market; the result of a search (i.e., the solution of a search) is referred to as an investment. Moreover, the solution space is divided into a set of subspaces called regions. Based on the concept of investment, a searcher is an agent who determines which region is worth being invested; that is, worth being searched.

B. The Concept

Most of the metaheuristics search for the solution without information of the solution space; that is, in a way just like a traveler who does not carry a map and has no methods to find the right way. Although without map to guide the traveler to the right direction, he or she still has a chance to reach the destination. However, the traveler may have a "very small chance" to reach the destination, especially when the geographical environment is complicated and large. By a very small chance, it means that the event of reaching the destination is incredibly unlikely to happen. In the very rare case, the traveler may still reach the destination after trial and error for a long time. Nevertheless, we just do not know "how much unnecessary long way" the traveler has traveled. The traveler may *never* find the right way to reach the destination until he or she gives up. This is just like we still have a chance to make money from a haphazard investment even if we do not understand the situation of the market. The bottom line is that we do not waste too much money on an investment, which can be regarded as the computing resource that has to be invested in the right region in the solution space so as to find the optimal solution of an optimization problem in question.

The basic idea of the proposed algorithm is that *you get whatever solution for which you actually pay*. The idea is to reduce the number of redundant searches in the search process, thus making each search of metaheuristics as meaningful as possible, instead of just a wild guess in the traditional way of search. That is why the search of metaheuristics on the solution space is regarded as an investment in this study. This concept implies that how the search of metaheuristics is performed and how the candidate solutions are measured will be very different. Here are some of the concerns. First comes an important information for search—the number of times a region is searched. This means that if the objective values of solutions of two different regions are the same, their

potentials may still be different because one region may have been searched for many more times than the other region. This is just like mining. Spending 100 days finding the gold which is worth 100 U.S. dollars in a region is very different from spending 1 hour finding the gold which is also worth 100 U.S. dollars in the same region. Second comes that the region which has not been searched for a long time will have a higher chance to find better solutions than the other regions which have been searched frequently. Third comes how to make metaheuristics work in parallel computing environment. Therefore, the parameters and information of the proposed algorithm cannot be centralized. In response to a parallel computing environment where the computing resources (nodes) may be increased or decreased dynamically, the design of the proposed algorithm also takes into account this implementation issue. Each computer node will be assigned a searcher and a region so that each computer node can be used to search the candidate solutions and record the searched information. This means that the proposed algorithm can increase the number of search directions and record more searched information as the number of computer nodes increases. By using this design, even though each computer node of SE does not have global information, it can still be used to search the solution just like it has the global information because the searched information will be exchanged between searchers and regions.

As shown in Fig. 2, assume we have four computer nodes (resources), the proposed algorithm will create four searchers each of which are associated with an investment s_i (i.e., solution). The solution space (called the market in this study) will then be divided into four regions each of which have two possible goods (i.e., candidate solutions). Considering these factors, it is the perspective of investment and information of marketing research that will be used in the design of the proposed algorithm.

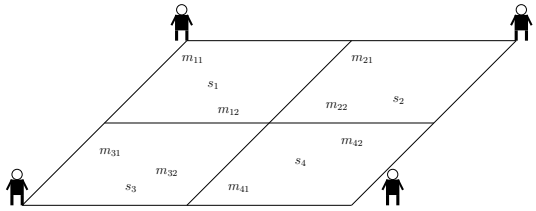

Fig. 2. The basic idea of SE.

Fig. 3 shows that the proposed algorithm consists of three main operators. The Resource_Arrangement operator plays the role of assigning the searchers to the regions of the market; the Vision_Search operator the role of searching; and the Marketing_Research operator the role of keeping track of information of each region for the search operators.

C. Resource Arrangement

As shown in Fig. 4, the Resource_Arrangement operator (RA) will divide the market (the solution space) into h regions. RA will first create w possible goods (candidate solutions) for each region r_j, all randomly, and then find the best good r_j^b

```
1  Search Economics()
2  {
3      s = Initialization()                                  I
4      m = Resource_Arrangement(s, r)
5      While the termination criterion is not met
6          s = Vision_Search(s, m)                           T, E, D
7          m = Marketing_Research(s, m)
8      End
9      Output s
10 }
```

Fig. 3. Outline of the proposed algorithm.

of each region r_j. The i-th searcher will randomly choose a good in the i-th region to invest, which means that at the very beginning of the search, the i-th searcher will be assigned to the i-th region if $n = h$; otherwise, some of the regions may be assigned either more than a searcher randomly or none at all.

```
1  Resource_Arrangment(s, r)
2  {
3      r = Divide the solution into h regions
4      For j = 1 to h
5          For k = 1 to w
6              m_jk = Random(r_j)
7          End
8          r_j^b = Best(m_j)
9      End
10     For i = 1 to n
11         s_i = Assign(m)
12     End
13 }
```

Fig. 4. Outline of the resrouce arragnment operator.

D. Vision Search

```
1  Vision_Search(s, m)
2  {
3      v = Transition(s, m)                                  T
4      e = Expected_Value(v, m, t^a, t^b)                    E
5      s = Determination(v, e)                               D
6  }
```

Fig. 5. Outline of the vision search operator.

As shown in Fig. 5, the Vision_Search operator (VS) is composed of three sub-operators to transit, evaluate, and determine the solution. Basically, this operator is similar to a traditional metaheuristic algorithm, implying that it can be replaced by a traditional metaheuristic algorithm as long as the evaluation and determination operators of the traditional metaheuristic algorithm can be adapted to fit with the spirit of the proposed algorithm. In this study, the transition operator is borrowed from the genetic algorithm for exchanging the information between the solutions; that is, in this study, the transition operator is composed of the crossover and mutation operators of the genetic algorithm. Unlike the genetic algorithm for which the information exchange is between chromosomes in the same population, the information exchange in the

proposed algorithm is between the "investment" of searchers and the "goods" of regions. Since there are n searchers and w goods (candidate solutions) each region, the transition operator will create a set of temporary candidate solutions v where v_{jk}^i is obtained by exchanging the information between the investment of the i-th searcher s_i and the k-th good of the j-th region m_{jk}.

Unlike the traditional metaheuristics for which the fitness of a solution is evaluated in terms of the fitness or objective value, for SE, the fitness of the good of the i-th searcher at the j-th region is evaluated in terms of the expected value e_{ij} defined by

$$e_{ij} = f_1(\mathcal{M}_j) f_2(\mathcal{V}_j^i) f_3(\rho_j). \tag{1}$$

In brief, Eq. (1) says that the expected value e_{ij} is composed of three pieces of information each of which is as described in turn below.

1) the status of the investment at the j-th region defined by

$$f_1(\mathcal{M}_j) = \frac{t_j^b}{t_j^a}, \tag{2}$$

which is used to measure the return of an investment at a particular region of the market. In other words, it is aimed at reducing the redundant searches in a region or to avoid falling into local optimum for a long time;

2) the objective value of the i-th searcher defined by

$$f_2(\mathcal{V}_j^i) = \frac{\sum_{k=1}^{w} f(v_{jk}^i)}{w}, \tag{3}$$

which is a measure of the potential of the investment of the i-th searcher at the j-th region based on the temporary candidate solutions in v.

3) the proportion or weight of the objective value of the best so far solution of each region of the market defined as

$$f_3(\rho_j) = \frac{f(m_j)}{\sum_{j=1}^{h} f(m_j)}. \tag{4}$$

In this study, the determination operator of SE is the tournament operator of the genetic algorithm but with a minor change. That is, the i-th searcher has to decide which region to invest although it is associated with the i-th region initially. Since the proposed algorithm is designed from scratch not to centralize all the information, in addition to v_{ii}, the determination operator will also randomly choose some of the v_{ij} for $i \neq j$ to enter the tournament process and then choose the better one to be the investment of the i-th searcher.

E. Marketing Research

As shown in Fig. 6, the Marketing_Research operator (MR) contains two kinds of operations: (1) update the market; i.e., keep information of the solutions that have been checked, and (2) accumulate the values of t^a and t^b. For the Update(s, m) operator, the main idea is to record everything about the solutions that have been checked to improve the search performance. However, there is simply no way to save all the searched solutions, i.e., the search history, because no matter how large it is, the memory space is limited. The trade-off the update operator takes is to save as much the information of the searched solutions (investments) as it can to achieve this goal. In this study, we try to keep k goods (candidate solutions) in each region. When finding a better good ($v_{jk}^i > r_j^b$) in region r_j, the update operator will use this good to replace one of the current goods in the same region. One possible solution to this problem is to compress the searched solutions into a single solution. In this case, the means of the searched solutions can be used to represent the searched solutions of SE which is defined as

$$\pi_{jk} = (\pi'_{jk} \times |\pi'_{jk}| + \sum_{z \in \pi_{jk}} z) \times \frac{1}{|\pi_{jk}|}, \tag{5}$$

where π_{jk} represents the k-the group of the j-th region that contains $|\pi_{jk}|$ searched solutions (goods) from the very beginning to the current iteration; π'_{jk} the mean of the searched solutions from the first iteration to the last iteration; $|\pi'_{jk}|$ the number of searched solutions from the first iteration to the last iteration; z a searched solution. When m_{jk} is set to π_{jk}, it means that all the the goods can be used to keep the search experience from the first iteration to the current iteration. Another solution is to perform a local search on the i-th region first and then use the common structure of sub-solutions to represent a set of searched solutions. In addition to relying on the compression or geoinformatics techniques to keep track of the information of searched solutions, random sampling is a makeshift method before we find a novel method to keep track of all the information of the search process of SE.

```
1 Marketing_Research(s, m)
2 {
3     m = Update(s, m)
4     t^a = Accumulate1(s, m)
5     t^b = Accumulate2(s, m)
6 }
```

Fig. 6. Outline of the marketing research operator.

The other important operation of MR is the accumulate operator. As shown in Fig. 6, t_j^a denotes how much computing resource we have invested on the j-th region while t_j^b denotes how many times SE searches the other regions but not the j-th region. If a searcher invests the j-th region, then t_j^b will be set to 1 and $t_j^a = t_j^a + 1$. Then, SE can use Eq. (2) to measure the potential of each region. For example, if the objective value of good (solution) m_{11} equals the objective value of good m_{21} but $f_1(\mathcal{M}_1) > f_1(\mathcal{M}_2)$, then it stands for the fact that good m_{11} has a higher potential than good m_{21} because SE invests much more computing resource searching region 2 than searching region 1 (e.g., $t_1^a < t_2^a$) or SE has not been searching region 1 longer than searching region 2 (e.g., $t_1^b > t_2^b$). Based on this concept, SE can avoid searching a particular region in the search space; instead, it will search more the regions that have a higher potential to find a better solution.

F. Summary

It can be easily understood that the basic idea of SE is how to use the "limited computing resources" to search for the solution in a "huge solution space." Since the computing resources we have are generally less than the computation costs we have to spend if we want to check every possibile solution in the solution space, the priority is then to make each search as meaningful as possible. Overall, it means that although the basic idea of metaheuristic algorithm is a strategical guess, we have to think about how to avoid duplicate and irrelevant guesses (i.e., searches). It is something like a good marketing research that can help us increase the successful rate of investment. It also means that SE has to know why searching a particular region. In summary, SE will map the solution space (i.e., the marketing research of SE) first and then search regions in the solution space that have a higher potential than the other regions to make a search as meaningful as possible. This is the main idea of the proposed algorithm which is aimed at finding the solution by the SE with limited resources. Moreover, if SE gets more computation resource on the convergence process, it will invest the resource to search for regions which have a higher potential than the other regions by using Eq. (1).

III. Experimental results

A. Experimental Environment and Parameter Settings

The empirical analysis was conducted on a PC with 2.50GHz Intel Core 2 Quad CPU Q8300 and 4GB of memory using Fedora 15 running Linux 2.6.42.9-1.fc15.x86_64, and the programs are written in C++ and compiled using g++. The problem used in the evaluation of the performance of the proposed algorithm is the OneMax problem [16]. It is an optimization problem the goal of which is to maximize the number of one's in a bit string. That is, assume $x = \langle x_1, x_2, \ldots, x_N \rangle$ where $x_i \in \{0, 1\}$ is the string. Then, the goal is to maximize:

$$F(x) = \sum_{i=1}^{N} x_i. \tag{6}$$

Apparently, the optimal solution of this problem is a bit string with all ones. For instance, if $N = 5$, then the optimal solution is $x = \langle 1, 1, 1, 1, 1 \rangle$. The OneMax problem is used as the touchstone of the proposed algorithm SE, for it can be easily transformed to the other optimization problems. In this study, we compare the performance of simulated annealing (SA) [3], genetic algorithm (GA) [5], and search economics (SE). The parameter settings of SA are as follows. The number of neighbors is set equal to 7. The parameter settings of GA are as follows. The population size is set equal to 8; the crossover rate to 0.8; the mutation rate to 0.001. The tournament selection and one-point crossover are used as the determination and transition operators. The parameter settings of SE are as follows. The number of searchers n is set equal to 4; the population size h to 4; the number of possible goods (the number of samples of each region) w to 2; $t_j^a = 1$; and $t_j^b = 1$ initially, i.e., at the first iteration. Note that in this study, we assume $n = h$. Note that because the OneMax problem is not an NP-hard problem, the random sampling is used as the update operator of the MR of the proposed algorithm. Also note that some additional checks are added to the determination of the VS; that is, the searcher chooses the new investment if it is better than the current investment. Each experiment is carried out for 30 runs, and the number of iterations each run is set equal to 1,000. All the experimental results shown are the average of 30 runs.

B. Results

As shown in Table I, SE provides a better result than the other metaheuristics compared in this study. The results of the simple GA are denoted GA1 and GA2 in Table I, respectively. For GA1, the number of chromosomes is set equal to 8; the crossover rate to 0.8; and the mutation rate to 0.1. For GA2, the number of chromosomes is set equal to 20; the crossover rate to 0.8; and the mutation rate to 0.2. Although SA gives a better result than GA1, it does not mean that the performance of GA is worse than that of SA. According to our observation, this is because the performance of GA is affected by the number of chromosomes (which is too small in this case) and the crossover and selection operators; therefore, GA cannot find a better result for large OneMax problems, i.e., for $N = 500$ and $N = 1,000$ in this case. Although GA2 still cannot provide a better result than SA, the results show that the performance of GA can be improved by increasing the number of chromosomes during the convergence process.

TABLE I
Comparison of quality.

dataset	SA	GA1	GA2	SE
$N = 10$	10.00	10.00	10.00	10.00
$N = 50$	49.60	48.93	49.67	50.00
$N = 100$	96.47	93.67	96.27	100.00
$N = 500$	457.87	363.03	404.60	493.03
$N = 1,000$	897.70	633.13	701.83	937.77

The results of SE show that the proposed algorithm can provide a better result than the other two metaheuristics compared in this study. According to our observation, the proposed algorithm can prevent the search diversity from decreasing or moving towards particular search directions because the search directions depend not only on the objective or fitness values but also on the information of the solution space that is also taken into account by SE. Table I also shows that the proposed algorithm gives a result that is close to the optimal solution, using only a simple transition operator to exchange the information. This means that the search performance of the proposed algorithm can be enhanced by using the transition and determination operators of different metaheuristics.

C. Circumstances of Convergence

Fig. 7 shows the convergence of SE for three OneMax problems (problems with $N = 10$, 100, and $1,000$, respectively). Because the initial solutions are randomly generated, the distance between the solution found by SE to the optimal solution is started with 0.5 while 1.0 represents the solution

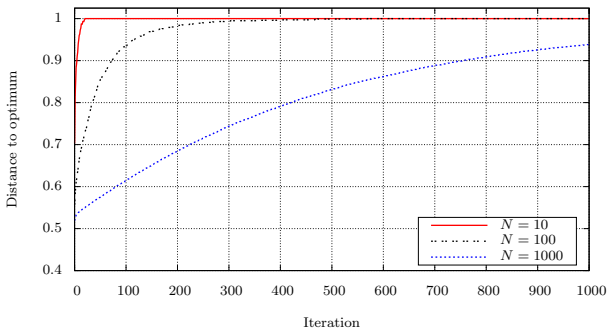

Fig. 7. The convergence of SE for the OneMax problem.

found by SE is exactly (i.e., 100%) the same as the optimal solution. From these results, it can be easily seen that the proposed algorithm can find an approximate solution or even the optimal solution if we invest enough computing resources. For example, in the case of the OneMax problem with $N = 10$ and $N = 100$, the proposed algorithm can find the optimal solution at iteration 20 and 570, respectively. Although SE cannot find the optimal solution of the problem with $N = 1,000$ within 1,000 iterations, the trends show that the results might be improved at later iterations.

That is why we conducted another simulation to further understand the performance of SE. The simulation results show that the proposed algorithm is almost capable of finding the optimal solution after 4,895 iterations on average. As a matter of fact, there is no guarantee that SE find the optimal solution every run; but the results show that it outperforms the other metaheuristics compared in this study. These results also show a distinguished feature of SE; that is, SE does not converge to a local optimum quickly. The reason is that the search diversity will not be degraded quickly even at the later iterations of the convergence process. That is, each region still has a good chance to be searched whereas the other metaheuristics may have converged to particular regions already. But this does not mean that SE can use only that kind of transition operator; rather, SE can actually use other transition operators to enhance its search performance for optimization problem.

IV. Conclusion

This paper presents an effective method, called search economics (SE), to enhance the performance of metaheuristic algorithm, by keeping track of information collected from the search process which includes not only the objective values of the candidate solutions but also the parts of solution space that has been explored. The solution presented in this study is to depict the solution based on the information we have in hand and the search algorithm can dynamically invest the computing resources to a region in the solution space that has a higher potential to find a better solution, meaning that SE will do its best to invest the computing resources it has to the regions where a better solution can be found. Since cloud computing or other parallel computing are the promising technologies, the proposed algorithm was designed from scratch with these issues in mind. As a result, by design, the proposed algorithm does not have the global information; instead, each searcher (computer node or worker) will play the role of keeping track of the solutions belonging to, and the best solution of, its region. With the addition of computing resources (computer nodes) to the search process, SE, by design, will divide a region into two new regions and then assign the searcher belonging to the original region and a new searcher to these two new regions to achieve the goal of parallel computing on the fly. In the future, our goal is to apply the SE to other optimization problems to show the performance of the proposed algorithm.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions on the paper. This work was supported in part by the Ministry of Science and Technology of Taiwan, R.O.C., under Contracts MOST103-2221-E-197-034 and MOST104-2221-E-197-005.

References

[1] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.
[2] F. Glover and G. A. Kochenberger, Eds., *Handbook of Metaheuristics*. Springer US, 2003.
[3] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[4] F. Glover and M. Laguna, *Tabu Search*. Kluwer Academic Publishers, 1997.
[5] J. H. Holland, *Adaptation in natural and artificial systems*. University of Michigan Press, 1975.
[6] M. Dorigo, V. Maniezzo, and A. Colorni, "The Ant System: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.
[7] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.
[8] R. Storn and K. Price, "Differential evolution—A simple and efficient adaptive scheme for global optimization over continuous spaces," TR-95-012, International Computer Science Institute, Tech. Rep., 1995.
[9] H. Abbass, "MBO: marriage in honey bees optimization-a Haplometrosis polygynous swarming approach," in *Proceedings of Computation Congress on Evolutionary Computation*, vol. 1, 2001, pp. 207 – 214.
[10] X. S. Yang, "A new metaheuristic bat-inspired algorithm," in *Proceedings of the Nature Inspired Cooperative Strategies for Optimization*, vol. 284, 2010, pp. 65–74.
[11] ——, "Firefly algorithms for multimodal optimization," in *Proceedings of the International Conference on Stochastic Algorithms: Foundations and Applications*, 2009, pp. 169–178.
[12] Z. W. Geem, J. H. Kim, and G. Loganathan, "A new heuristic optimization algorithm: Harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.
[13] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.
[14] S. Salcedo-Sanz, J. D. Ser, S. Gil-López, I. Landa-Torres, and J. A. Portilla-Figueras, "The coral reefs optimization algorithm: An efficient meta-heuristic for solving hard optimization problems," in *Proceedings of the Applied Stochastic Models and Data Analysis International Conference*, 2013, pp. 751–758.
[15] C. W. Tsai and J. Rodrigues, "Metaheuristic scheduling for cloud: A survey," *IEEE Systems Journal*, vol. 8, no. 1, pp. 279–291, 2014.
[16] J. Schaffer and L. Eshelman, "On crossover as an evolutionary viable strategy," in *Proceedings of the International Conference on Genetic Algorithms*, R. Belew and L. Booker, Eds. Morgan Kaufmann, 1991, pp. 61–68.