A Multiple-Search Multi-Start Framework for Metaheuristics for Clustering Problems

KAI-CHENG HU, CHUN-WEI TSAI, AND MING-CHAO CHIANG
Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan
Corresponding author: Ming-Chao Chiang (mcchiang@cse.nsysu.edu.tw)

This work was supported in part by the Ministry of Science and Technology of Taiwan under Contract MOST108-2221-E-110-028 and Contract MOST108-2221-E-005-021-MY3.

ABSTRACT
Metaheuristic algorithms have been widely used as an effective and efficient way for solving various complex optimization problems; there is, however, plenty of room for improvement. In this research area, the two most important issues that greatly influence the final results of single-solution-based metaheuristic algorithms are in that: (1) some of them are extremely sensitive to the initial solutions, and (2) some of them are easy to fall into a local optimum at early iterations. For these reasons, an effective framework, called multiple-search multi-start for single-solution-based metaheuristic algorithm (MSMS-S), is presented in this paper to mitigate the impact of these issues. MSMS-S ensures that a search procedure will be given different search directions based on the so-called re-start mechanism. To evaluate the performance of the proposed framework, we compare it with several well-known single-solution-based metaheuristic algorithms for clustering and codebook generation problems. Simulation results show that the proposed framework is capable of improving the performance of single-solution-based metaheuristic algorithms.

INDEX TERMS Multi-start, metaheuristic algorithms, and clustering.

I. INTRODUCTION
Metaheuristics [1], [2] have been developed for over half a century and have been successfully applied to a wide range of fields to solve difficult optimization problems, such as scheduling problems in a cloud computing environment [3], [4] and data mining problems for the internet of things [5], [6]. A well-known classification method for the metaheuristics is based on the number of candidate solutions (or search directions) searched at a time during the search process. If the algorithm searches one solution at a time, it is referred to as a single-solution-based algorithm (SSBA), such as simulated annealing [7] and tabu search [8]. On the contrary, if the algorithm searches multiple solutions at a time, it is referred to as a population-based algorithm (PBA), like genetic algorithm [9] and swarm intelligence [10].

All the SSBA and PBAs have pros and cons. One of the main advantages of an SSBA is that it is simple and easy to implement. However, searching one and only one solution at a time makes it easy to fall into a local optimum at early iterations and thus hard to escape from the local optimum to find a better solution. This somehow implies that most SSBA are extremely sensitive to the initial solution. That is, if the initial solution is not at a region that has potential to find good solutions, then it is very unlikely that an SSBA will find a good solution during the convergence process.

The main difference between SSBA and PBA is that PBA searches multiple directions at a time per iteration. This characteristic implies that an integral part of a PBA is a way to exchange information between the searched agents. If a new PBA is designed for an optimization problem but each search agent searches its solution in the solution space independently from each other during the convergence process, then it would look like that multiple SSBA are performed at the same time. This is the reason why PBA can find solutions near the optimum quicker than SSBA by using the same number of iterations. The exchange of information eventually also decreases the impact of the initial solutions. PBA is able to find better results than SSBA with the same number of evaluations in most cases because the information exchange mechanism provides a better chance to escape the local optimum and also leads the search direction to regions that may have better solutions during the convergence process. However, PBA also has several disadvantages that need to
be addressed. An obvious drawback is that it takes a longer computation time to converge [11] because most of the sub-solutions of a PBA reach the final state at different times. This implies that many of the computations during the convergence process of a PBA are eventually redundant. Another drawback is that PBA generally lacks the capability of local search.

Since every metaheuristic algorithm has its pros and cons, to enhance the influence of pros and reduce the effect of cons, hybrid metaheuristic algorithms [12] and hyper metaheuristic algorithms [13] represent two common ways to reach these goals by combining multiple metaheuristic algorithms. The underlying idea of these two kinds of metaheuristic algorithms is to leverage the strength of multiple metaheuristic algorithms during the convergence process to enhance the probability of finding good results. The main difference is in that the so-called hybrid metaheuristic algorithms mix the strategies of different metaheuristic algorithms at each iteration while the so-called hyper metaheuristic algorithms randomly choose a strategy from the metaheuristic algorithms used and perform it for a certain number of iterations. Although both of these two methods can improve the quality of the end results, there still are problems that need to be solved. The main problem of a hybrid metaheuristic algorithm is in that it generally takes much more computation time than the original ones at each iteration. Another problem is how to balance exploration and exploitation during the convergence process for both the hybrid and hyper metaheuristic algorithms, especially when there are a large number of local optimal solutions in the search space. In addition to hybrid and hyper metaheuristic algorithms, another way to improve the performance of a metaheuristic algorithm is the so-called multi-start method [14], which provides an approach that is similar to the hybrid and hyper metaheuristic algorithms in the search of the optimal solution during the convergent process. The main idea of a multi-start method is to repeat the search algorithm several times with different initial solutions. These different initial solutions give the search algorithm a better chance to search different directions. However, the multi-start method also has problems that should be addressed, such as how to exchange information among candidate solutions and when to re-start the search process.

This study is built on the multiple-search multi-start framework (MSMS) [15] to balance the exploration and exploitation of SSBAs when the search process gets stuck in local optima of certain regions. A key point to reach this goal is to find a way to pass the searched information among the search directions at different iterations. The framework described herein, called multiple-search multi-start for single-solution-based metaheuristic algorithms (MSMS-S), is presented to enhance the performance of SSBAs in terms of the quality. The proposed framework takes a novel approach that makes it possible for an SSBA to take the pros of a PBA which can provide a better result than the SSBA by itself does while not increasing as much the computation time as the PBA does. As an extended version of [15], the main differences and contributions of this study can be summarized as follows:

1) In this study, we apply the proposed algorithm not only to the clustering problem but also to the codebook generation problem, to show the performance of the proposed algorithm.

2) Moreover, we add a more detailed description to explain how the proposed algorithm works and how to apply it to other optimization problems.

The remainder of the paper is organized as follows. In Section II, a brief review on the multi-start methods for metaheuristic algorithms and a summarization of the characteristics of these methods are given. The basic idea and the detailed strategies of the proposed framework are given in Section III. Section IV begins with a description of the simulation environment, followed by the simulation results of applying MSMS-S to $k$-means, simulated annealing (SA), tabu search (TS), particle swarm optimization (PSO), and generalized Lloyd algorithm (GLA) to show its performance and comparing it with the other two multi-start methods, namely, greedy randomized adaptive search procedure (GRASP) and iterated local search (ILS). Also discussed in this section are further research issues. Finally, the conclusion and some future research directions are drawn in Section V.

II. RELATED WORK

Most of the multi-start algorithms are designed for the same purpose; that is, to improve the search diversity of the search process and to avoid falling into and thus getting stuck in local optima. Re-starting the search process is a simple way to improve the search diversity. The re-start mechanism will redirect the search process to a non-explored region in the search space and then start the search process [14] again. Another way for improving the search diversity is to increase the search directions by using multiple start points (initial positions) during the convergence process [16]. In [14], [16], several different perspectives, such as systematic/randomized, memory/memory-less, rebuild/build, and single solution/multiple solutions, are used to categorize multi-start methods. According to Marti’s observation [14], simple multi-start procedures based on metaheuristic algorithms to solve the combinatorial optimization problem have been presented in many studies. Fig. 1 gives an outline of the multi-start method. A later study [17] pointed out that multi-start methods are usually divided into two phases; namely, phase to generate a solution and phase to improve the solution. After that, the multi-start methods will update the best solution if the new solution is better than the best solution.

For a multi-start algorithm, how the searched information is passed around to different stages of the search algorithm has been a critical issue that has been addressed by several studies [18], [19]. The greedy randomized adaptive search procedure (GRASP) [18] is one of the methods presented to deal with this issue. As a well-known multi-start method,
Set $i = 1$

While (the stopping condition is not met)

1. **The generation phase**

   $x_i = \text{Construct}\_\text{Solution}();$

2. **The search phase**

   $x_i' = \text{Search}(x_i);$

   If ($x_i'$ is better than $b$)

   $b = x_i'$

   $i = i + 1;$

End

**FIGURE 1. Outline of the multi-start algorithm.**

---

1. **GRASP**

   For $i = 1$ to $N$

   1. **The generation phase**

      $x = \text{Construct}\_\text{Solution}();$

   2. **Evolutionary local search (ELS)**

      While (the stopping condition is not met)

      1. **The search phase**

         $x_i' = \text{Search}(x_i);$

         If ($x_i'$ is better than $b$)

         $b = x_i';$

      End

   End

**FIGURE 2. Outline of the hybrid multi-start algorithm.**

GRASP has its own strategy to pass information among several search procedures and uses the so-called restricted candidate list (RCL) to keep track of the information of the best candidates in the generation phase. GRASP will then use the subsolutions of the best candidates in the RCL to generate the new solution randomly. In this way, the results of different search procedures can be shared. This kind of multi-start method keeps some of its experiences by controlling the generation phase so that its experiences can be passed to different search phases. The algorithm begins with the search at different start positions in the search phase based on the information passed to it and is aimed to find a better result. In other words, GRASP plays the role of a high-level search method to generate the solutions and determine the search directions.

As shown in Fig. 2, Duhamel et al. [20] adopted the evolutionary local search (ELS) [21] as the low-level search method and GRASP as the high-level search method to control the timing of re-start. In Fig. 2, $N$ denotes the number of iterations of the GRASP algorithm. Ali and Gabere [22] combined simulated annealing (SA) and a multi-start algorithm in which SA is used to determine if the newly generated start point is accepted. If it is accepted, SA will invoke the local search procedure with this point as the start point to converge to a new local optimum. Yepes et al. [23] used threshold accepting (TA) as the low-level algorithm of GRASP to determine the maintenance programs of pavements in which a threshold is used to decide whether or not to accept the solution found by the neighborhood search. The threshold is high initially but will decrease as the number of iterations increases. This strategy allows the search directions to converge at early iterations but gives a higher diversity at later iterations.

Another issue about keeping the search solution in memory is addressed by ILS [19] which presents a solution with a history-based perturbation and uses the result of the previous search procedure to generate the new start position of another search procedure. An acceptance criterion is also used here to decide the timing of updating the best solution. In [24], three kinds of heuristic algorithms are used to create new initial solutions, and ILS with a tabu list is used to reach the local optimum. The tabu list avoids repeatedly searching the same region for a while. Once the ILS procedure is finished, it will reinitialize the solution with three heuristic algorithms and start the ILS procedure again. In [25], a similar multi-start algorithm was presented in which a randomized constructive heuristic method is used to generate the initial solution, and a local search with perturbation is used to search the objectives. Two parameters are used to control the number of times the local search procedure is performed and the number of times the search phase cannot find better solutions to dynamically determine when to terminate the search phase. Silva et al. [26] combined ILS and randomized variable neighborhood descent (RVND) [27] to solve the split delivery vehicle routing problem (SDVRP).

In summary, a common characteristic of multi-start method is that it can improve the search diversity and has a higher probability of finding a better result. Besides, it can be easily applied to most metaheuristic algorithms [14] although there exist issues that need to be addressed. One of the issues is, how the searched information is kept and exchanged in the search process? Another is, when and how often does an algorithm have to re-start or re-initialize the search procedure? Still another is the issue of how to balance exploitation and exploration during the convergence process. This issue is important not only in most multi-start methods but also in most metaheuristic algorithms. Re-starting the search process or re-initializing the start points of the search procedure too frequently will make the algorithm act like a random search. On the other hand, if the re-start interval is long, the search process may get stuck in some regions easily, thus wasting time doing some redundant computations. Our observation shows that these issues imply that all the multi-start algorithms need an effective mechanism to find the suitable timing for re-start to prevent the search procedure from struggling at certain regions, which will waste time doing a lot of redundant computations. Furthermore, the multi-start methods also need to distribute the suitable search space to each search procedure and transmit the information among these procedures to improve the probability of finding a better local optimum, or even the global optimum.

III. THE PROPOSED FRAMEWORK

**A. THE CONCEPT**

Compared to PBAs, the pros and cons of SSBAs are obvious—less time of computation but lower quality of result. Therefore, the basic idea of the proposed framework (MSMS-S) is to focus on making an SSBBA perform like a PBA without taking too much computation time. A good
choice to reach this goal is the so-called multi-start-based mechanism, which can increase the diversity of a search procedure while taking just a little more computation time. This concept makes it possible for the search procedure to have a higher diversity to avoid falling into local optima easily, but not to diverge too much. MSMS-S keeps some of the subsolutions of the ancestors to provide information that may lead to a better result while being perturbed a little bit at the same time to have more different search directions. This concept is essentially the main distinction between the MSMS-S framework and all the other multi-start algorithms. In addition, MSMS-S can be easily applied to all metaheuristic algorithms instead of just some particular metaheuristic or some specific optimal problems. MSMS-S can be seen as a high-level framework in which metaheuristic algorithms are used as the low-level search method.

**B. DETAILS OF MSMS-S**

The proposed framework is as shown in Fig. 3. The notation used is as follows:

| Variable | Description |
|----------|-------------|
| \( p \)  | The candidate pool. |
| \( x \)  | The solution. |
| \( x' \) | The solution searched starting from \( x \). |
| \( n_s \) | The number of stages. |
| \( m \)  | The size of \( p \). |

As can be easily seen from Fig. 3, the proposed framework contains five operators; namely, Construct_Solution(), Inbreeding_Solution(), Search(), Add2Pool(), and Select(). Construct_Solution() is used to generate the initial solution, and Inbreeding_Solution() is used to generate a solution with perturbation. Multiple_Search() is the main search procedure composed of Search() and Add2Pool(). In Multiple_Search(), Search() will find a solution by using a low-level SSBA, and Add2Pool() will put the solution into the candidate pool.

Moreover, the proposed framework consists of \( n_s \) stages each of which has a candidate pool\(^1\) of size \( m \) and so is the number of substages. Each solution, denoted \( x = \{x_1, x_2, \ldots, x_n\} \), contains \( n \) subsolutions. The first step of the proposed framework is to first empty the candidate pool \( p \) and then call Construct_Solution() to generate the initial solution \( x \) randomly. At the beginning of each stage, Multiple_Search\((x, p)\) will find a new solution \( x' \) by calling Search\((x)\) and putting \( x' \) in the candidate pool by calling Add2Pool\((x', p)\). Then, for each substage, Inbreeding_Solution\((p)\) will create a new initial solution \( x \) using subsolutions in \( p \) and perturb it a bit to make it a bit different from those in the candidate pool. A parameter called inheritance ratio is used here to decide the percentage of information that will be used to create the new initial solution. For example, if the inheritance ratio is set equal to 50%, it means that 50% of the subsolutions of \( x \) are taken from the subsolutions in \( p \) while the other 50% of the subsolutions of \( x \) are randomly generated by perturbation. This operator makes it possible not only to exchange the searched information in \( p \) but also to search different directions using perturbation at the same time. As shown in Fig. 4, the inbreeding process of the proposed algorithm will randomly select a subsolution from a searched solution in the pool or randomly create a new subsolution to construct a new solution. This example shows that the first, second, and fourth subsolutions of the new solution are taken from the searched solutions in the pool while the third and fifth subsolutions are created by a random process. Therefore, the new solution \( x \) will inherit some of the characteristics of the searched solutions in the pool but not exactly the same because a certain percentage of subsolutions are created by a random process which can be regarded as the perturbation process.

Then, Multiple_Search\((x, p)\) will use the new initial solution \( x \) to find another solution \( x' \) and put it in \( p \). If the candidate pool \( p \) is full, it means that the current stage ends. Before the new stage begins, the candidate pool \( p \) will be emptied right after Select\((p)\) saves the best solution in \( p \) as the new initial solution for the first substage of the next stage. This strategy can not only enhance the diversity of the search process of the next stage but also use the so-far-best solution to make the search direction of the next stage not diverge too much.

In the proposed framework, Multiple_Search\((x, p)\) is an important operator which is used to control, and record the result of, the low-level metaheuristic algorithm. Various search strategies can be used in this operator. This means

\(^1\)The candidate pool can be considered as the population of a PBA the solutions of which will be filled in by the substages of each stage.

**FIGURE 4.** A simple example for illustrating how Inbreeding_Solution() works.
that as far as the proposed framework is concerned, different search strategies (e.g., different search algorithms, such as k-means and simulated annealing) can be used in the Search(x) operator to solve different optimization problems. Even more, different search algorithms can be applied to different solutions although one and only one (meta)heuristic algorithm is used in this study. In other words, this is how the low-level metaheuristic algorithm is run to search for better solutions from the start point \( x \) to improve solutions which have already been found. It can be a complete run of the algorithm or just one-one iteration search procedure. Then, the Add2Pool(\( x' \), \( p \)) operator will put the solution \( x' \) into the candidate pool \( p \). It can also use different strategies to record the search result. For example, the simplest way is to put the complete solution into the pool, or it can put a subsolution of the result into the pool, or even a solution with some perturbations. The strategies can be designed to fit the problem which the proposed framework is used to solve. The Select(\( p \)) operator is used to select a solution from the pool \( p \) and the selected solution will be the new initial solution of the next stage. The Select(\( p \)) operator used in the proposed framework simply selects the best solution in \( p \). It can certainly use different strategies to select a solution that is more suitable to the problem in question.

In this study, the terms stage and substage are used to describe the search process of the MSMS-S framework. In brief, the whole search process consists of a certain number of stages each of which consists of a certain number of substages as far as the proposed framework is concerned, and all the substages of a stage will share the same candidate pool. The searched information of the previous substage can be passed to the next substage by saving it to the pool. These procedures make it possible for each stage to inherit the search experiences of the previous stage. These experiences can enhance the search diversity, thus preventing the search process from converging to particular regions quickly. The number of substages in a stage is determined by the size of the candidate pool. When the search process is going to start a new stage, all the information in the pool will be cleared except the best one. This strategy will give the search process a higher probability to search different regions. The termination criterion of MSMS-S can be a predefined number of substages, stages, iterations, or evaluations, or it can even be something else.

C. A SIMPLE EXAMPLE
To make it easier to understand the MSMS-S framework, Fig. 5 gives a simple example to illustrate how it works. This example shows how the initial solution for each substage is generated and how the result for each substage is added into the candidate pool. It can be easily seen from Fig. 5 that the candidate pool size is assumed to be four; so is the number of substages, thus the number of solutions to be generated each stage. For the very first substage \( \phi_{11} \), the initial solution is generated randomly (denoted by the empty squares) because the candidate pool is empty at this moment. For all the other substages, the initial solution is generated partially based on the solutions found and saved in the pool by the previous substages and partially randomly. This is why the initial solution of the second substage contains empty squares and squares with number 1 inside. This process will be repeated for the second substage the solution of which is denoted by squares with number 2 inside, the third substage the solution of which is denoted by squares with number 3 inside, and so on, all the way up until the last substage of a stage is run and the result is obtained and saved.

Once the candidate pool is full, the procedure will move on to the second stage. However, before starting the second stage, the candidate pool will be flushed to empty once the initial solution for the first substage of the second stage, which is \((4, 4, 1, 3)\) in this case, is generated. MSMS-S will now move on to the second stage repeating exactly what it did in the first stage. Then, once the second stage is completed, it will move on to the third stage, and so on until all the stages are completed or the termination criterion is met.

IV. EXPERIMENTAL RESULTS
A. EXPERIMENTAL ENVIRONMENT AND DATASETS
All the experiments are conducted on an Intel i7 CPU 920 with 2.67 GHz CPU and 4GB of memory running Fedora 12 with Linux 2.6.31.5-127. Also, the programs are written in C and compiled by gcc version 4.4.2 20091027 (Red Hat 4.4.2-7). To evaluate the performance of the proposed framework, we apply it to the clustering [28] and codebook generation problems [29].

For the clustering problem, all the ten datasets, as detailed in Table 1, are taken from UCI [30], and all of them are normalized except “Iris” because the fitness values of the original Iris data can help us check the accuracy.
of the results. As for the codebook generation problem, the twelve images—namely, airplane, baboon, boat, bridge, clown, crowd, girlface, Lena, pepper, man, Saturn, and Zelda the size of which are as shown in Table 2—are downloaded from: http://extras.springer.com/2016/978-3-319-22303-2/SoftN1/Store and http://www.hlevkin.com/06 testimages.htm. All the images used in this experiment are in the format of 8-bit grayscale bitmap (.bmp) and are available for download from the following website: http://oslab.cse.nsysu.edu.tw/vq-pic/. Moreover, the codebook size is set equal to 64; and the block size to 4 × 4 pixels.

### B. SIMULATION RESULTS OF CLUSTERING PROBLEM

To evaluate the performance of the proposed framework for the clustering problem, in this paper, the MSMS-S framework was applied to three single-solution-based metaheuristics algorithms; namely, k-means (KM), simulated annealing (SA), and tabu search (TS). All the algorithms are carried out for 100,000 evaluations each run, and the averages of 30 runs are taken as the results, as shown in Tables 3 and 4. This means that every search algorithm compared in this study will check the same number of possible solutions each run. The initial temperature of SA is set equal to 100.0 degrees. The size of the tabu list of TS is set equal to 10. For the PSO-based algorithms, the inertia factor ω is set equal to 0.8; the cognitive ratio c₁ to 1.5; the social earning ratio c₂ to 1.5; and the population size to 10. The setting of parameters of MSMS-S is as follows: the pool size is set equal to 10, and the inheritance ratio is set equal to 90%. As we mentioned in Section III-B, the number of stages is set equal to the population size; i.e., 10.

The sum of squared errors (SSE) defined as

\[
SSE = \sum_{i=1}^{k} \sum_{j=1}^{n_i} \|x_{ij} - c_i\|^2,
\]

where \( k \) is the number of clusters; \( n_i \) the number of data points in the \( i \)-th cluster; \( x_{ij} \) the \( j \)-th data point in the \( i \)-th cluster; and \( c_i \) the centroid of the \( i \)-th cluster is used as the measure of the results described in Table 3. SSE can be seen as a kind of intra distance measurement. The smaller the value, the closer the nodes are to the centroids of their groups. For clustering, this implies a better result. The results described in Table 4 are in terms of time in seconds.

It can be easily seen from Table 3 that MSMS-S is able to enhance the quality of the search results in most cases. This is mainly due to the fact that the MSMS-S framework can give a higher diversity during the convergence process, thus providing SSBA a better chance to search different regions to get a better result, especially when the search procedure gets stuck in a local optimum. On the other hand, a higher diversity may get you a worse result when it comes to the cases in which a higher convergent ability is needed, such as abalone and yeast. For these complex datasets, it is hard for the low-level metaheuristic algorithms to find the local optimum in the current region. In this case, the MSMS-S framework will re-start the search procedure to search another region even though the low-level algorithm has not yet finished the search of the current region. This makes it possible for MSMS-S to get a worse result than the original version.

The results depicted in Table 4 show that the algorithms with the MSMS-S framework take more time than the original algorithms in some cases. The MSMS-S framework does not use complex strategies to maintain the search diversity. However, the search strategies of a low-level algorithm make it possible for the algorithm to take more time at the early iterations of the search procedure than at the later iterations, say, 2 seconds during the first 100 iterations but 1 second during the last 100 iterations. The reason is that more solutions are changed at early iterations but less at later iterations during the convergence process. In the MSMS-S framework, the low-level algorithm will start a new search procedure after re-starting. For this reason, a larger number of re-starts implies a longer computation time.

### C. SIMULATION RESULTS OF CODEBOOK GENERATION PROBLEM

To evaluate the performance of the proposed framework for different optimization problems, in this study, the proposed framework was also applied to particle swarm optimization (PSO) [31] and generalized Lloyd algorithm (GLA) [32] for solving the codebook generation problem (CGP). All the algorithms are carried out for 10,000 iterations each run, and the averages of 30 runs are taken as the results. As for the setting of the parameters of MSMS-S, the pool size is set

| Table 1. Datasets for the clustering problem. |
|-----------------------------------------------|
| dataset | # of patterns | # of clusters | # of attributes |
|--------|--------------|---------------|-----------------|
| DSC1   | Abalone      | 4,177         | 29              | 8               |
| DSC2   | Balance-scale| 569           | 2               | 30              |
| DSC3   | Ecoli        | 336           | 8               | 7               |
| DSC4   | Haberman     | 306           | 2               | 3               |
| DSC5   | Iris         | 150           | 3               | 4               |
| DSC6   | Liver        | 345           | 2               | 6               |
| DSC7   | SPECT        | 267           | 2               | 22              |
| DSC8   | SPECTF       | 267           | 2               | 44              |
| DSC9   | Wine         | 178           | 3               | 13              |
| DSC10  | Yeast        | 1,464         | 10              | 8               |

| Table 2. Size of images for the codebook generation problem. |
|-------------------------------------------------------------|
| Name  | Size (in pixels)   |
|-------|--------------------|
| DSG1  | airplane           | 512 × 512       |
| DSG2  | baboon             | 512 × 512       |
| DSG3  | boat               | 512 × 512       |
| DSG4  | bridge             | 512 × 512       |
| DSG5  | clown              | 512 × 512       |
| DSG6  | crowd              | 512 × 512       |
| DSG7  | girlface           | 512 × 512       |
| DSG8  | Lena               | 512 × 512       |
| DSG9  | man                | 1024 × 1024     |
| DSG10 | pepper             | 512 × 512       |
| DSG11 | Saturn             | 1480 × 1200     |
| DSG12 | Zelda              | 768 × 512       |

Note that some of the images have been adjusted for the experiment described herein.
equal to 10, and the inheritance ratio is set equal to 90%. For the GLA-based algorithms, the number of evaluations is set equal to 5,000. For the PSO-based algorithms, the inertia factor $\omega$ is set equal to 0.8, the cognitive ratio $c_1$ and the social learning ratio $c_2$ are both set equal to 1.5, and the population size is set equal to 10.

The CGP is to partition a set of input patterns $X = \{x_1, x_2, \ldots, x_n\}$ in $d$-dimensional space into a certain number of groups, denoted $\Pi = \{\pi_1, \pi_2, \ldots, \pi_m\}$, each of which is associated with a codeword $c_i$ the collection of which is called a codebook, denoted $C = \{c_1, c_2, \ldots, c_m\}$, where $c_i$ and $\pi_i$ are defined as

$$c_i = \frac{1}{|\pi_i|} \sum_{x \in \pi_i} x,$$  

$$\pi_i = \{x \in X \mid \|x - c_i\| \leq \|x - c_j\|, \forall j \neq i\},$$

with $X = \bigcup_{i=1}^{m} \pi_i$ and $\pi_i \cap \pi_j = \emptyset$, $\forall i \neq j$. The CGP can be easily reduced to the clustering problem by simply mapping the $d$-dimensional patterns of the CGP to the $d$-attribute patterns of the clustering problem. This means that an algorithm for solving the clustering problem efficiently can be used to solve the CGP efficiently. To measure the quality of a solution of the CGP, a commonly used metric is the peak signal-to-noise ratio (PSNR) defined as

$$\text{PSNR} = 10 \times \log_{10} \frac{255^2}{\text{MSE}},$$

where $255$ is the peak value of a gray-level image, and MSE is the mean squared error defined as

$$\text{MSE} = \frac{1}{h \times w} \sum_{i=1}^{h} \sum_{j=1}^{w} (\hat{v}_{ij} - v_{ij})^2,$$

where $v_{ij}$ and $\hat{v}_{ij}$ are the pixel values at row $i$ and column $j$ of the input image and the reconstructed image, respectively. A higher value of PSNR implies that the compressed image looks more like the original image; that is, the loss of the signals is less.

It can be easily seen from Table 5 that the proposed framework can improve the PSNR of GLA and PSO in all cases. This implies that the proposed framework can also enhance the performance of algorithms aimed to solve the CGP.

Table 6 shows the results in terms of the computation time. In most cases, the MSMS version of all the methods takes more time than the original version. The proposed framework requires additional time for generating the new initial solutions, especially when a PBA that has a large number of...
solutions to be generated is used as the low-level algorithm. On the contrary, the MSMS version of GLA takes less time than the original version in some cases. The reason comes from the details of the implementation. In brief, the underlying idea is for GLA to stop the search of solution as soon as the local optimum has been found, so is MSMS + GLA except that MSMS + GLA will first save the local optimum into the pool and then re-start a new search. In this way, any further computations that are essentially redundant (i.e., that cannot improve the quality of the solution) can be reduced. One of the important capabilities of the proposed framework is that it can speed up the convergence of a search algorithm to local optima quickly. In these cases, MSMS + GLA takes a fewer number of iterations than GLA to reach the local optima, and the time saved is more than the time spent generating new solutions. For this reason, MSMS + GLA takes less time than GLA.

In summary, the results show that the proposed framework has potential to solve different kinds of problems. It can enhance the search diversity of an SSBA to make it have a higher chance to jump from one region to another to find out a better result. Similarly, it can also improve the performance of a PBA to make it possible to find solutions that are closer to the local optima. On the other hand, although it takes a little bit extra time to generate the initial solutions for each low-level algorithm, the proposed framework can sometimes even speed up the convergence of the search procedure. In some cases, when an SSBA is used as a low-level search algorithm, the time saved from eliminating redundant computations is larger than the time taken for generating the initial solutions. This makes a method with MSMS perform better than its original version.

### D. COMPARISON WITH OTHER RE-START METHODS

In this section, we compare MSMS-S with two famous metaheuristic algorithms with re-start mechanism; namely, greedy randomized adaptive search procedure (GRASP) and iterated local search (ILS). All the algorithms are carried out for 100,000 evaluations each run, and the averages of 30 runs are taken as the results, as shown in Tables 7 and 8. MSMS-S, GRASP, and ILS all use $k$-means as the low-level search algorithm because the problem to be solved is the clustering problem as far as this paper is concerned. Moreover, for this experiment, the RCL size of GRASP is set equal to 10.

|                  | MSMS-S+KM | GRASP | ILS |
|------------------|-----------|-------|-----|
| DSC1             | 34.27     | 34.16 | 34.20 |
| DSC2             | 3472.32   | 3481.61 | 3472.44 |
| DSC3             | 14.40     | 15.41 | 14.60 |
| DSC4             | 25.28     | 25.32 | 25.29 |
| DSC5             | 78.95     | 87.63 | 78.94 |
| DSC6             | 30.00     | 30.19 | 30.00 |
| DSC7             | 1005.44   | 1008.91 | 1005.13 |
| DSC8             | 214.20    | 214.29 | 214.20 |
| DSC9             | 48.95     | 48.98 | 48.96 |
| DSC10            | 45.35     | 47.25 | 45.86 |

The pool size and inheritance ratio of MSMS-S are set equal to 10 and 90%, respectively.

The numbers in bold indicate the algorithm that gives the best result for the dataset evaluated. Again, Table 7 uses SSE and Table 8 uses the computation time as the measurement. It can be easily seen from Tables 7 and 8 that MSMS-S can provide a better search result than GRASP and ILS in terms of the quality. As for the computation time, MSMS-S was beaten by both ILS and GRASP, but only marginally. In brief, this can be easily justified by the fact that the MSMS-S framework leverages the strength of GRASP and ILS to maintain a higher diversity during the convergence process so that it has a higher probability to search different regions, thus finding a better result. On the other hand, to maintain a higher search diversity, the strategies used by MSMS-S are more complex than GRASP and ILS. Especially, the generation of new initial solutions from the candidate pool and the update of the candidate pool at the beginning of each stage both take extra computation time.

### E. IMPACT OF PARAMETER SETTINGS AND CONVERGENCE ANALYSIS

In this section, the impact the setting of the parameters of MSMS-S—pool size and inheritance ratio—may have on the quality of the clustering results is analyzed. Both experiments described here (one for the pool size; the other for the inheritance ratio) use the $k$-means algorithm, tabu search, and simulated annealing as the low-level search algorithms. Moreover, both are carried out for 30 runs each of which performs 100,000 evaluations, and the averages of 30 runs are taken as the results.

#### 1) CANDIDATE POOL SIZE

In this experiment, the inheritance ratio is set equal to 90%. The influence of pool size is as shown in Fig. 6. The size of the candidate pool decides the number of search results that can be kept. Obviously, a larger pool implies that more information from the previous search procedures can be retained. The setting of pool size will also impact the lifetime of stages. When the pool size is large, the period for flushing the pool is long. This implies that the generation of new solutions will refer to the similar group of search results for a long time. It can be easily seen from this experiment that the effect of the pool size on the quality of the search results is not as apparent
for ecoli and yeast as for abalone for which a smaller pool size implies a better result. This is due to the fact that abalone contains more attributes and has more clusters than the other two datasets and implies that the variations in the terrain of the search spaces of abalone is more complex. A smaller pool makes it possible in the generation of new solutions to have a higher chance to refer to the so-far-best result, which will in turn intensify the low-level search algorithm to search regions that have high quality of solutions. The experimental results show that a small pool size is useful in the case that the dataset in question has a large number of attributes and clusters.

2) INHERITANCE RATIO
In this experiment, the pool size is set equal to 10. The influence of the inheritance ratio, which is defined as the percentage of information in the generation of new solutions when the search process re-starts comes from the pool, is as shown in Fig. 7. A 0% inheritance ratio means that all the new solutions are generated randomly while a 100% inheritance ratio means that all the new solutions are generated using the information in the pool. It can be easily seen from Fig. 7 that the best results occur when the inheritance ratio is 100%. The influence of inheritance ratio is apparent for the abalone dataset, but not for the ecoli dataset. As shown in Table 1, the datasets for which the influence is apparent have more attributes and more clusters. This implies that the terrains of the search spaces of these datasets are more complex; thus, a search algorithm needs more information to guide its search directions to obtain better results. As far as the MSMS-S framework is concerned, the higher the inheritance ratio, the more the past experience is used in the generation of new solutions. This makes it likelier for the low-level search algorithms to spend more time searching regions that have a better result.

3) CONVERGENCE ANALYSIS
Fig. 8 shows the convergence results of all the algorithms compared in this study to evaluate their capability in solving the clustering problem. It can be easily seen that the proposed algorithm is capable of enhancing the search ability of a clustering algorithm from the very beginning to the very end of the search. For example, for the three datasets tested, not only can MSMS+PSO find a better result than PSO alone, it can even keep improving the result at later stage. More precisely, it can be easily seen from Fig. 8(b), even after 60,000 evaluations, MSMS+PSO can still improve the clustering results in terms of SSE. Of course, the similar results can also be found by applying the proposed algorithm to other clustering algorithms, such as $k$-means, TS, and SA. In summary, the results show that the proposed algorithm is capable of enhancing the performance of the clustering algorithms in most cases.
F. DISCUSSIONS

The MSMS-S framework can enhance the quality of the search result of the low-level metaheuristic algorithms. However, the total computation time will be influenced by the search strategies and parameter settings of the low-level algorithms. For KM, the computation time of each iteration remains the same; thus, the total computation time of MSMS-S+KM is the sum of the computation time of KM and the MSMS-S framework. As for TS and SA, the computation time of each iteration varies. The MSMS-S framework can also provide a better search result than other re-start methods. However, MSMS-S takes longer computation time than the other re-start methods. This is basically a trade-off between the quality of results and the computation time.

For applications the focus of which are on the quality of the results, MSMS-S is more suitable than GRASP and ILS. The experimental results show that the influence of the pool size is not obvious. Besides the strategies taken by MSMS-S, a possible reason is that the influence of this parameter is not obvious for the clustering problem. This assumption needs to be verified by applying MSMS-S to other kinds of problems. The inheritance ratio needs to be set according to the complexity of datasets or problems. Generally speaking, for a simple dataset and problem, the inheritance ratio should be set equal to a larger value. However, for a more complex dataset and problem, the inheritance ratio should be set equal to a smaller value. Besides testing each parameter separately, another issue worth observing is the influence between the pool size and inheritance ratio. A reasonable assumption is that the information in the candidate pool seems to be less important for an inheritance ratio that is set equal to a small value. More experiments are needed to verify this assumption.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed an efficient algorithm, called multiple-search multi-start for single-solution-based metaheuristic algorithm (MSMS-S), to enhance the performance of single-solution-based metaheuristic algorithms for clustering. The re-start mechanism and the concept of multiple search are used to increase the diversity of the search procedures. Besides, it adds a candidate pool to keep track of the solution found by the low-level algorithms. While MSMS-S re-starts the search procedure of a low-level algorithm, it will use information in the candidate pool to generate the new initial solution. This strategy can avoid starting the search procedure from worse positions while at the same time keeping the diversity of the search procedure, thus improving the quality of the search result. The experimental results show that the single-solution-based metaheuristic algorithms with MSMS-S framework can get better results than their original versions in most cases. The results further show that the quality enhancement of MSMS-S is better than all the other state-of-the-art re-start methods evaluated in this paper.

One of our future goals is to apply MSMS-S to more metaheuristics to evaluate its robustness. Another goal is to develop strategies to adjust the parameter settings used by the MSMS-S framework itself and to combine different low-level algorithms instead of using the same algorithm in each run. We will also develop strategies to control the diversity of the search procedure more efficiently. Finally, we will apply MSMS-S to problems in real life, especially the big data problems and the applications of internet of things (IoT).

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions on the article.

REFERENCES

[1] F. Glover and G. A. Kochenberger, Handbook Metaheuristics. New York, NY, USA: Springer, 2003.
[2] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization: Overview and conceptual comparison,” ACM Comput. Surv., vol. 35, no. 3, pp. 268–308, Sep. 2003.

[3] C.-W. Tsai and J. J. P. C. Rodrigues, “Metaheuristic scheduling for cloud: A survey,” IEEE Syst. J., vol. 8, no. 1, pp. 279–291, Mar. 2014.

[4] S. T. Milan, L. Rajabion, H. Ranjarb, and N. J. Navinipour, “Nature-inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments,” Comput. Oper. Res., vol. 110, pp. 159–187, Oct. 2019.

[5] C.-W. Tsai, “Data mining for Internet of Things: A survey,” IEEE Comm. Surv. Tuts., vol. 16, no. 1, pp. 77–97, 1st Quart., 2014.

[6] V. Ghezavati and P. Hosseini, “Application of efficient metaheuristics to solve a new bi-objective optimization model for hub facility location problem considering value at risk criterion,” Soft Comput., vol. 22, no. 1, pp. 195–212, Jan. 2018.

[7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” Science, vol. 220, no. 4598, pp. 671–680, 1983.

[8] F. Glover and M. Laguna, Tabu Search. Norwell, MA, USA: Kluwer, 1997.

[9] J. H. Holland, Adaptation in Natural and Artificial Systems. Ann Arbor, MI, USA: Univ. Michigan Press, 1975.

[10] A. P. Engelbrecht, Fundamentals of Computational Swarm Intelligence. Norwell, MA, USA: Wiley, 2006.

[11] C.-W. Tsai, S.-P. Tseng, M.-C. Chiang, and C.-S. Yang, “A framework for accelerating metaheuristics via pattern reduction,” in Proc. 12th Annu. Conf. Genet. Evol. Comput., 2010, pp. 293–294.

[12] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli, “Hybrid metaheuristics in combinatorial optimization: A survey,” Appl. Soft Comput., vol. 11, no. 6, pp. 4135–4151, Sep. 2011.

[13] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu, “Hyper-heuristics: A survey of the state of the art,” J. Oper. Res. Soc., vol. 64, no. 12, pp. 1695–1724, Dec. 2013.

[14] R. Martí, “Multi-start methods,” in Handbook Metaheuristics, vol. 57, F. Glover and G. A. Kochenberger, Eds. New York, NY, USA: Springer, 2003, pp. 355–368.

[15] C.-W. Tsai, K.-C. Hu, and M.-C. Chiang, “A multiple-search multi-start framework for metaheuristics,” in Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC), Oct. 2014, pp. 2774–2779.

[16] A. Gyorgy and L. Kocsis, “Efficient multi-start strategies for local search algorithms,” J. Artif. Intell. Res., vol. 41, pp. 407–447, Jul. 2011.

[17] R. Martí, M. G. C. Resende, and C. C. Ribeiro, “Multi-start methods for combinatorial optimization,” Eur. J. Oper. Res., vol. 226, no. 1, pp. 1–8, Apr. 2013.

[18] T. A. Feo and M. G. Resende, “Greedy randomized adaptive search procedures,” J. Global Optim., vol. 6, no. 2, pp. 109–133, 1995.

[19] H. R. Lourenço, O. C. Martin, and T. Stützle, “Iterated local search,” in Handbook Metaheuristics, vol. 57, F. Glover and G. A. Kochenberger, Eds. New York, NY, USA: Springer, 2003, pp. 320–353.

[20] C. Duhamel, P. Lacomme, A. Quilliot, and H. Toussaint, “A multi-start evolutionary local search for the two-dimensional loading capacitated vehicle routing problem,” Comput. Oper. Res., vol. 38, no. 3, pp. 617–640, Mar. 2011.

[21] S. Boettcher and A. G. Percus, “Extremal optimization: An evolutionary local-search algorithm,” in Computational Modeling and Problem Solving in the Networked World (Operations Research/Computer Science Interfaces Series), vol. 21, H. K. Bhargava and N. Ye, Eds. New York, NY, USA: Springer, 2003, pp. 61–77.

[22] M. M. Ali and M. N. Gabere, “A simulated annealing driven multi-start algorithm for bound constrained global optimization,” J. Comput. Appl. Math., vol. 233, no. 10, pp. 2661–2674, Mar. 2010.

[23] V. Yepes, C. TORRES-MACHI, A. Chamorro, and E. Pellicer, “Optimal pavement maintenance programs based on a hybrid greedy randomized adaptive search procedure algorithm,” J. Civil Eng. Manage., vol. 22, no. 4, pp. 540–550, Aug. 2016.

[24] V.-P. Nguyen, C. Prins, and C. Prodhon, “A multi-start iterated local search with tabu list and path relinking for the two-echelon location-routing problem,” Eng. Appl. Artif. Intell., vol. 25, no. 1, pp. 56–71, Feb. 2012.

[25] J. Michallet, C. Prins, L. Amodeo, F. Yalaoui, and G. Vité, “Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services,” Comput. Oper. Res., vol. 41, pp. 196–207, Jan. 2014.

[26] M. M. Silva, A. Subramanian, and L. S. Ochi, “An iterated local search heuristic for the split delivery vehicle routing problem,” Comput. Oper. Res., vol. 53, pp. 234–249, 2015.

[27] P. H. V. Penna, A. Subramanian, and L. S. Ochi, “An iterated local search heuristic for the heterogeneous fleet vehicle routing problem,” J. Heuristics, vol. 19, no. 2, pp. 201–232, Apr. 2013.

[28] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: A review,” ACM Comput. Surv., vol. 31, no. 3, pp. 264–323, Sep. 1999.

[29] N. M. Nasrabadi and R. A. King, “Image coding using vector quantization: A review,” IEEE Trans. Commun., vol. 36, no. 8, pp. 957–971, 1988.

[30] UCI Datasets. Accessed: Mar. 29, 2020. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Iris

[31] J. Kennedy and R. C. Eberhart, “Particle swarm optimization,” in Proc. IEEE Int. Conf. Neural Netw., Mar. 1995, pp. 1942–1948.

[32] S. Lloyd, “Least squares quantization in PCM,” IEEE Trans. Inf. Theory, vol. 28, no. 2, pp. 129–137, Mar. 1982.

KAI-CHENG HU received the M.S. and Ph.D. degrees in computer science and engineering from National Sun Yat-sen University, Kaohsiung, Taiwan, in 2011 and 2019, respectively. His research interests include metaheuristic, evolutionary algorithm, machine learning, and artificial intelligence.

CHUN-WEI TSAI received the Ph.D. degree in computer science and engineering from National Sun Yat-sen University, Kaohsiung, Taiwan, in 2009. He was a Postdoctoral Fellow with the Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, before joining the Faculty of the Applied Geoinformatics and the Information Technology, Chia Nan University of Pharmacy and Science, Tainan, Taiwan, in 2010 and 2012, respectively. He joined the Faculty of the Department of Computer Science and Information Engineering, National Ilan University, Yilan, Taiwan, in 2014, the Department of Computer Science and Engineering, National Chung-Hsing University, Taichung, Taiwan, in 2017, and then the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, in 2019, where he is currently an Assistant Professor. His research interests include computational intelligence, data mining, cloud computing, and the Internet of Things. He has been serving as the Senior Associate Editor for the Journal of Internet Technology, since 2014, the Associate Editor for IEEE Access and IET Networks, since 2017 and 2018. He has been a member of the Editorial Board of the International Journal of Internet Technology and Secured Transactions (IJITST) and the Journal of Network and Computer Applications (IJNCA), since 2016 and 2017, respectively.

MING-CHAO CHIANG received the B.S. degree in management science from National Chiao Tung University, Hsinchu, Taiwan, 1978, and the M.S., M.Phil., and Ph.D. degrees in computer science from Columbia University, New York, NY, USA, in 1991, 1998, and 1998, respectively. He has over 12 years of experience in the software industry encompassing a wide variety of roles and responsibilities in large and start-up companies before joining the Faculty of the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, in 2003, where he is currently a Professor. His research interests include image processing, evolutionary computation, and system software.