Hybrid Metaheuristics and their Implementations

http://dx.doi.org/10.3991/ijoe.v11i7.4762

Caichang Ding[1] , Wenxiu Peng[1] and Weiming Wang[2]
[1] School of Computer Science, Yangtze University, Jingzhou 434023, China
[2] Teaching Supervision Office, Shenzhen Institute of Information Technology, Shenzhen 518172, China

***Abstract*—This paper studies with the design of hybrid metaheuristics and their implementations. Hybrid metaheuristics involve some major issues that could be classified as design and implementation. Combining different kinds of methods is a ordinary strategy to solve optimization problems. As we have developed a unified view of metaheuristics, that is based on their key search components, one can say that designing a multi-objective metaheuristic can be reduced to select the most suited search components and combining them. Each of these metaheuristics has been proven successful on a variety of applications. Although there have been attempts to compare their performance, the results are contradicting and inconclusive. A difference is made between the design issues used to introduce hybridization and implementation issues that depend on the execution model of the algorithms.**

***Index Terms*—hybrid metaheuristics, model, design, implementation.**

I. INTRODUCTION

In the past decades, a multitude of new search heuristics, often called "metaheuristics" have been proposed, many of them inspired by principles observed in nature. Common representatives include evolutionary algorithms (EAs) [1], ant colony optimization (ACO) [2], simulated annealing [3], tabu search [4], or estimation of distribution algorithms [5]. Each of these metaheuristics has been proven successful on a variety of applications. Although there have been attempts to compare their performance, the results are contradicting and inconclusive. There does not seem to be a superior candidate that should generally be preferred over the others. Thus, it is not surprising that recently, there has been a growing interest in hybridization of these metaheuristics.

Hybridization of metaheuristics involves a few major issues that may be classified as design and implementation. The former category is concerned with the hybrid algorithm itself, involving issues such as functionality and architecture of the algorithm. The implementation consideration includes the hardware platform, programming, model, and environment on which the algorithm is to be run. In this paper, a difference is made between the design issues used to introduce hybridization and implementation issues that depend on the execution model of the algorithms.

II. DESIGN PRINCIPLES

Given a description of the different metaheuristics in general form has many benefits. First, it creates a common language, which allows researchers from different fields to understand each other's approaches easily. Second, it moves the focus from the complete algorithms to the components. Third, it provides the interfaces for the different components to work together. Based on the presented unified framework, it is almost straightforward to combine different components from different algorithmic paradigms: an algorithm designer can easily select a combination of memorization features, choose a suitable set of construction operators or create new ones that make use of the combined set of selected memorization features, and then decide how the memory is updated with the newly generated information. The framework allows for a lot of freedom: new solutions may be constructed in different ways, using different information from the memory, the solutions thus constructed using one part of the memory may be used to update another part of the memory, and so on.

The taxonomy will be kept as small as possible by proceeding in a hierarchical way as long as possible, but some choices of characteristics may be made independent of previous design choices, and thus will be specified as a set of descriptors from which a subset may be chosen. The taxonomy proposed here is a combination of these two schemes: hierarchical as long as possible to reduce the total number of classes and flat when the descriptors of the algorithms may be chosen in an arbitrary order.

III. HIERARCHICAL CLASSIFICATION

A discussion about the hierarchical portion then follows. At the first level, one may distinguish between low-level and high-level hybridizations. The low-level hybridization addresses the functional composition of a single-optimization method. In this hybrid class, a given function of a metaheuristic is replaced by another metaheuristic. In high-level hybrid algorithms, the different metaheuristics are self-contained. There is no direct relationship to the internal workings of a metaheuristic. In relay hybridization, a set of metaheuristics is applied one after another, each using the output of the previous as its input, acting in a pipeline fashion. Teamwork hybridization represents cooperative optimization models in which many cooperating agents evolve in parallel; each agent carries out a search in a solution space. Three classes are derived from this hierarchical taxonomy.

A. LRH (low-level relay hybrid)

This class of hybrids represents algorithms in which a given metaheuristic is embedded into a S-metaheuristic. Few examples of hybrid metaheuristics belong to this class.

B. LTH (low-level teamwork hybrid)

As mentioned in Chapter 1, two competing goals govern the design of a metaheuristic: exploration and exploitation. Exploration is needed to ensure that every part of the space is searched enough to provide a reliable estimate of the global optimum. Exploitation is important since the refinement of the current solution will often produce a better solution. P-metaheuristics (e.g., evolutionary algorithms, scatter search, particle swarm, ant colonies (AC)) are powerful in the exploration of the search space and weak in the exploitation of the solutions found.

Therefore, most efficient P-metaheuristics have been coupled with S-metaheuristics such as local search, simulated annealing, and tabu search, which are powerful optimization methods in terms of exploitation. The two classes of algorithms have complementary strengths and weaknesses. The S-metaheuristics will try to optimize locally, while the P-metaheuristics will try to optimize globally. In LTH hybrid, a metaheuristic is embedded into a P-metaheuristics. This class of hybrid algorithms is very popular and has been applied successfully to many optimization problems. Most of the state-of-the-art P-metaheuristics integrate into S-metaheuristics.

C. HRH (high-level relay hybrid)

In HRH hybrids, the self-contained metaheuristics are executed in a sequence. For example, the initial solution of a given S-metaheuristic may be generated by another optimization algorithm. Indeed, the initial solution in S-metaheuristics has a great impact on their performances. A well-known combination scheme is to generate the initial solution by greedy heuristics, which are in general of less computing complexity than iterative heuristics.

This scheme may also be applied to P-metaheuristics, but a randomized greedy heuristic must be applied to generate a diverse population. Greedy heuristics are in general deterministic algorithms and then they generate always the same solution. On the other hand, the diversity of the initial population has a great impact on the performance of P-metaheuristics. This hybrid scheme is carried out explicitly in the scatter search metaheuristic.

Combining P-metaheuristics with S-metaheuristic in the HRH scheme is also largely applied. It is well known that P-metaheuristics are not well suited for fine-tuning structures, which are very close to optimal solutions. Indeed, the strength of P-metaheuristics is in quickly locating the high-performance regions of vast and complex search spaces. Once those regions are located, it may be useful to apply S-metaheuristics to the high-performance structures evolved by the P-metaheuristics. A fundamental practical remark is that after a certain amount of time, the population is quite uniform and the fitness of the population is no longer decreasing. The odds to produce fitter individuals are very low. That is, the process has fallen into a basin of attraction from which it has a low probability to escape.

The exploitation of the already found basin of attraction to find as efficiently as possible the optimal point in the basin is recommended. It is experimentally clear that the exploitation of the basin of attraction that has been found may be more efficiently performed by another algorithm than by a P-metaheuristics. Hence, it is much more efficient to use a S-metaheuristic such as a hill-climbing or tabu search. The HRH hybridization may use a greedy heuristic to generate a good initial population for the P-metaheuristics. At the end of a simulated annealing search, it makes sense to apply local search on the best found solution to ensure that it is a local optimum.

IV. COMBINING EVOLUTIONARY ALGORITHMS AND ANT COLONY OPTIMIZATION

In this section, we propose a number of EA/ACO hybrids, which attempt to combine the two memorization schemes. Before that, however, let us briefly present the pure EAs [1] and ACO [2] we built on. The application considered is the traveling salesperson problem (TSP) [6].

Maybe the most straightforward combination of EAs and ACO is to simply use both basic algorithms to generate a portion of the new solutions each. More specifically, in every cycle, we generate 50% of the k new solutions on the basis of the pheromone matrix, while the remaining 50% are generated using the edge recombination operator. The complete set of k new solutions is then used in the standard way to update the pheromone matrix as well as the population.

Pheromone Completion (PC) crossover: This operator at the same time uses pheromone matrix and population to create a new individual. First, an individual is selected from the population by rank-based selection. A random connected part of that individual is then chosen, and this partial permutation is completed using the standard ACO construction operator, that is probabilistically according to the pheromone matrix. After k individuals have been created that way, the new individuals are used to update the pheromone matrix as well as the population. Note that this operator is somewhat similar to the approach suggested by Punch. However, while we are proposing to use separate memory structures for the population and the pheromone matrix, they propose to use a population of agents, each agent consisting of a solution and an individual pheromone matrix. It is difficult to reason whether using a global pheromone matrix or individual pheromone matrices is more promising. A global pheromone matrix collects more information and will, therefore, perhaps be a better guide in particular in short runs. Individual pheromone matrices, on the other hand, allow for different solutions to be encoded simultaneously, which may be more beneficial for long optimization runs, when diversity is a major issue.

Pheromone-supported Edge Recombination (PsER) crossover: It may happen that all the four edges used by the edge recombination operator to select the next city lead to cities that have already been visited. In these cases, edge recombination selects a city randomly. The pheromone-supported edge recombination operator suggested here instead uses the probabilistic selection based on the pheromone matrix. Again, the resulting individual is subject to mutation, and after k individuals

have been created, all of them are used to update both the pheromone matrix as well as the population.

Mutating Ants (MutA): ACO maintains diversity by choosing cities probabilistically in every step, and thus does not seem to require additional mutations. However, a simple change, like swapping two cities is very unlikely to be produced by the probabilistic construction procedure. Therefore, we here suggest to mutate the solutions created by the ACO, adding a different kind of change.

Ant-based crossover (ABX): The idea of this hybrid is to combine ACO's sequential construction and elegant integration of heuristic knowledge with the population-based memory of EAs. ABX selects parents from the population just as an ordinary EA. These parents are used to construct a temporary pheromone matrix, by initializing all pheromones to the same small value, and allowing the parents to place additional pheromone on the solutions (paths) they represent. New solutions are then constructed in an ACO way, based on the temporary pheromone matrix and possible heuristic knowledge. Only the best of the generated solutions is returned as child and used to update the population (memory). It is straightforward to extend this idea by running an ACO on the temporary pheromone matrix for a few iterations (allowing the generated solutions to update the temporary pheromone matrix before generating some more solutions). We simply used the same parameter settings that had shown to be successful in that paper: two parents are selected, and the ACO is run on the temporary matrix for 5 iterations with 12 ants each. Only the best solution found is returned as a child.

V. EXPERIMENTAL RESULTS

Since ACO is primarily designed for permutation problems, we chose a simple symmetric Euclidean TSP with 100 cities to compare the different algorithms. Three problem instances of varying difficulty have been created: in problem P1, all cities are located equally spaced on a unit circle. To generate problem instances P2 and P3, the location of each of P1's city has been moved in a random direction by a distance of 0.2 or 0.5, respectively. Independent of the problem instance, each algorithm was allowed to create and evaluate 200,000 solutions.

As expected, heuristic domain knowledge is able to drastically improve performance. The ACO with heuristic knowledge as well as ABX generate equally good (presumably optimal) results on all problem instances, outperforming all methods without heuristic knowledge. Note that besides the idea of ABX, incorporation of domain knowledge into the EA is not as straightforward, and we have not been able to produce similar results, for example, by seeding the population with a heuristic. As the results show, the problem instances examined are too simple if heuristic knowledge is incorporated.

VI. CONCLUSIONS

The main drawback of hybridization is the introduction of new parameters that define the hybrid scheme. The setting of these parameters is nontrivial. A crucial question that has to be addressed in the future is an aid for the efficient design of hybrid metaheuristics in which the automatic setting of parameters must be investigated. Indeed, it will be interesting to guide the user to define the suitable hybrid scheme to solve a given problem. It will also be interesting to define adaptive cooperation mechanisms that allow to select dynamically the optimization methods according to convergence or other criteria such as diversity. Some approaches such as hyper heuristics have been proposed to deal with this problem. These approaches are dedicated to choose the right heuristic for the right operation at the right time during the search. It must be noted that these hybrid approaches operate in the heuristic space, as opposed to most implementations of metaheuristics that operate in the solution space. This principle is relatively new, although the concept of optimizing heuristics is not a recent one.

For the algorithm designer, of course, it would be invaluable to know which operators and memory schemes are most promising depending on the application at hand. However, that assumes a useful categorization of problems, and is thus several steps in the future. Overall, we hope that this paper helps to gain a general understanding of different metaheuristics and of the way they interact.

ACKNOWLEDGMENT

This work is supported by Scientific Research Projects of Hubei Provincial Department Of Education. The authors are also grateful to the four anonymous referees for their insightful and constructive comments, which greatly improved the quality of the paper.

REFERENCES

[1] J. H. Holland. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Harbor, MI, 1975.

[2] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1: 53–66, 1997. http://dx.doi.org/10.1109/4235.585892

[3] E. Aarts and J. Korst. Simulated Annealing and Boltzmann Machines. John Wiley & Sons, New York, 1989.

[4] F. Glover. Tabu search. ORSA Journal of Computing, 1: 190–206, 1989. http://dx.doi.org/10.1287/ijoc.1.3.190

[5] P. Larrañaga and J.A. Lozano, Eds. Estimation of Distribution Algorithms. Kluwer Academic, New York, 2002. http://dx.doi.org/10.1007/978-1-4615-1539-5

[6] K. L. Hoffman, M. Padberg and G. Rinaldi. Traveling salesman problem. In Encyclopedia of Operations Research and Management Science, 1573-1578, Springer, 2013. http://dx.doi.org/10.1007/978-1-4419-1153-7_1068

AUTHORS

Caichang Ding received the B.Sc. degree from the School of Mechanical & Electronic Information, China University of Geosciences, Wuhan, China, in 2003, the M.Sc. degree from the School of Computer, Wuhan University, Wuhan, China, in 2006, and the Ph.D. degree at State Key Laboratory of Software Engineering, Wuhan University, Wuhan, China, in 2014. He is currently a lecturer in the School of Computer Science, Yangtze University, Jingzhou, China. His main research interests include computational learning theory, statistical learning, basic theory of evolutionary computation and optimization theory. (e-mail: ccding_2006@hotmail.com).

Wenxiu Peng received the B.Sc. and M.Sc. degrees from Hubei University, Hubei, China, in 2003 and 2006. She is currently a lecturer in the School of Computer Science, Yangtze University, Jingzhou, China. Her main research interests include computational learning theory, statistical learning, basic theory of evolutionary computation and optimization theory. (e-mail: dianxin_1999@qq.com).

Weiming Wang received the B.Sc. degree from the School of Machinery and Automation, Wuhan University of Science & Technology, Wuhan, China, in 2003, and the M.Sc. and Ph.D. degrees from the School of Computer, Wuhan University, Wuhan, China, in 2005 and 2010, respectively. He is currently a lecturer at Teaching Supervision Office, Shenzhen Institute of Information Technology, Shenzhen, China, Wuhan. His main research interests include computational learning theory, statistical learning, basic theory of evolutionary computation and optimization theory. (e-mail: 173676358@qq.com).

Submitted 25 May 2015. Published as resubmitted by the author 25 june 2015.