International Journal of
Science and Engineering Investigations vol. 1, issue 8, September 2012
ISSN: 2251-8843

Population-Based Metaheuristics: A Comparative Analysis

Elena Simona Nicoară
Information Technology, Mathematics and Physics Department, Petroleum-Gas University of Ploieşti, Ploieşti, Romania
(snicoara77@yahoo.com, snicoara@upg-ploiesti.ro)

Abstract- To optimally solve hard optimization problems in real life, many methods were designed and tested. The metaheuristics proved to be the generally adequate techniques, while the exact traditional optimization mathematical methods are prohibitively expensive in computational time. The population-based metaheuristics, which manipulate a set of candidate solutions at a time, have advantages over the single-state methods and therefore are preferred techniques when hard problems are to be solved. Such metaheuristics include Genetic Algorithms, Ant Colony Optimization, Particle Swarm Optimization, Scatter Search and many more methods. In this survey a comparative analysis of the main population-based metaheuristics was accomplished; the focus is on the fundamental properties regarding operational principle, on the adequate problems, the advantages and disadvantages in use.

Keywords- *metaheuristic; candidate solution; optimization.*

I. INTRODUCTION TO METAHEURISTICS

A wide class of real world problems is optimization problems. In economics (corporate finance, investments, production, distribution, purchasing, human resources), in industry (mechanics, engineering, airlines and trucking, oil and gas, electric power), in agriculture, in financial services and many other fields, optimality is a central issue. We try to maximize the profit, the flow, the resistance, the efficiency, the utility, or to minimize the cost, the time spent, the loss, the risks etc. while some constraints are satisfied. To note that many times, multiple antagonist objectives have to be simultaneously fulfilled.

The first methods proposed to solve optimization problems were the exact optimization techniques: the traditional mathematical optimization methods (direct methods and gradient methods), the enumerative methods (guided or unguided), Lagrangian relaxation and decomposition strategies. These optimization methods are designed to find all the global optimal solutions to the problems. A very good characteristic! Why need other optimization techniques?

When little information about the problem is known, or the complexity level is high, or the instance dimensions are big, all the beautiful mathematical tools become mostly unhelpful. There are many real world problems that are multimodal, nonlinear, deceptive and multi-objective. The search space for such problems, if additionally the instances are big, is huge and hard to explore. Finding all exact solutions for such problems may be possible with traditional mathematical methods, in certain conditions, but it takes a prohibitively long time to find the solution.

Deb reported in [1] the main disadvantages of exact methods: lack of global perspective over the problem, general inefficiency for discrete variable problems, tendency to be blocked to suboptimal solutions when search space is difficult, convergence dependency on the initial solution, inefficiency in parallel computing environments. All these drawbacks restrain applicability of exact techniques to solving small dimensions optimization instances.

This fact led to designing approximate optimization techniques and made of heuristics theory a research area with fast extent, especially in the last forty years. Heuristic algorithms significantly reduce computational time to solve hard problems while return good enough solution(s), eventually the global optimal one(s).

Initially, problem-specific heuristics were proposed to supplement standard optimization algorithms. Then, many research studies tried to unify heuristic mechanisms adequate to many problems in (almost) general heuristic algorithms. This is the case for heuristic algorithms for combinatorial optimization for example.

The term "metaheuristic", first used by Glover in [2], is used to designate a complex iterative heuristic algorithm, with the aim of efficiently explore and exploit the search space in order to solve hard problems. Metaheuristics are the most adequate algorithms when [3]:

- few useful information is known about the problem:
  - we don't know how the optimal solution looks like;
  - we don't know how to approach finding it in a principled way;
  - very little heuristic information about the problem exists and brute-force search is out of the question because the search space is too large;
- but if a candidate solution to the problem is provided, one can test it and evaluate how good it is.

Many real world problems fit into this framework. An exemple is to find an optimal structure of 15 values for the components of some robot (dimensions of arms, type of

material for the arms and joints, density for some materials etc.). To explore all the points in the 15-dimensions search space is unpractical. Moreover, the optimality can imply multi-objective functions: overall resistance, walk speed, manufacturing and maintainance cost, dimensions etc. Another example of such hard instance is finding an optimal placement of 50 scrap collection centers in a vast area (over 10 000 houses), when the distances between houses and collection centers and housing density determine optimality.

Additional reasons for the solving difficulty are present: the problems often cannot be formulated and solved by a direct mathematical approach, many constraints and many parameters are involved, the number of possible solutions can be enormous, qualitative solutions must be fast identified, verifying all the possible solution to find the best one is very expensive (often unfeasible), because the quality of a solution may vary in time many solutions are necessary.

To solve such hard problems, the most metaheuristics make use of randomness to guide search through the search space. This is the reason why metaheuristics is the major subfield of stochastic optimization, or stochastic search in a more general approach. Also, the term "black box optimization" is sometimes associated to metaheuristics, the reason being the lack of data about the problem.

Though the scope is mostly optimization, other fields which benefit from metaheuristics are prediction, classification and pattern recognition.

II. POPULATION-BASED METAHEURISTCS

Metaheuristics can be classified in many ways: on the existence of memory component, on the number of candidate solutions manipulated in each iteration, on the parallelization ability, on the dynamics of objective function, on the extent of search (local or global search), on the integrality of used candidate solutions (incremental construction or complete candidate solutions), on the problem class which are adequate to [4]. The number of candidate solutions which are manipulated in every iteration, the most applied criterion, splits the metaheuristics into two major classes:

- Single-state metaheuristics and
- Population-based metaheuristics (population methods).

A single-state metaheuristic uses a single candidate solution, complete or partial, at a time. This is updated on specified events, and the algorithm runs until the end condition is met. The best candidate solution is returned as optimal solution. Examples of single-state metaheuristics are: Hill-climbing, Iterated Local Search, Simulated Annealing, Tabu Search and GRASP (Greedy Randomized Adaptive Search Procedure).

A population-based metaheuristic manipulates a collection of candidate solutions at a time. These possible solutions generally are not independent (meaning that they are not parallel hill-climbers): they influence the way the others candidate solutions close with the optimum. The main advantage of a population method against the single-state metaheuristic results from the definition - in final stage, many (near) optimal solutions can be returned. Moreover, if difficult search space is the case, good solutions are with high probability not lost and new candidate solutions with special characteristics are to be generated from other solutions in the collection.

The population-based metaheuristics include the vast class of Evolutionary Computation techniques and Scatter Search method. The Evolutionary Computation methods are naturally inspired computational techniques, based on concepts and mechanisms in biology, genetics, evolution and nature in general. They appear classified in two categories:

- *Evolutionary algorithms*: Genetic Algorithms, Evolution Strategy, Evolutionary Programming, Differential Evolution;
- *Swarm intelligence methods*: Ant Colony Optimization, Particle Swarm Optimization, Artificial Immune Systems, Bees Algorithm, Wasp Behavioral Model, Charged System Search, Cuckoo Search, Firefly Algorithm, Gravitational Search Algorithm, Intelligent Water Drops, Learning Classifier Systems, Cultural Algorithms, Self-organization methods etc.

All these metaheuristics were designed in the last decades; some of them in the last 5-10 years. Some are generalizations of single-state metaheuristics and some are derived from other population-based metaheuristics. Therefore, a comparative analysis made on the main methods, which constitutes skeletons in the class, is sufficient to obtain pertinent results. These principal methods are: Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Scatter Search (SS).

III. MAIN POPULATION-BASED METAHEURISTICS: SHORT DESCRIPTION

In the following, the above mentioned population-based metaheuristics are to be shortly described.

A. Evolutionary Algorithms. GAs

The evolutionary algorithms are metaheuristics based on applying in computer science principles derived from evolutionary biology: inheritance, genetic mutation, natural selection and crossover. In major cases, the nature follows two simple principles:

- If by genetic processes a descendant with above average performance is generated, it has a higher survival rate than an average individual and hence it has higher chances to produce descendants which inherit good traits than an average individual;
- If a below average performance individual is generated, it does not survive long enough and therefore it will be eliminated from the evolving population.

All the evolutionary algorithms (mentioned in the previous section) are based on evolutionary principles, but operate differently and are designed for different categories of problems. The distinction refers to the types of alterations required to candidate solutions when generate descendants, to the parents selection methods and to the data structures used in solution representation [5].

The notable characteristics of evolution strategies are the normal distributed mutation, the self-adapting parameters values and two populations existence - the parents' population and the offspring' population. In GAs, the focus is on genetic operators simulating the natural evolution. In evolutionary programming, on the other hand, the focus is put on the behavioral relation between parents and offspring.

GAs are the most common evolutionary algorithm, applied and tested on almost every type of adequate problem. They are parallel algorithms which transform a population of mathematical objects (possible solutions of the problem) in a new population using three operators similar to evolutionary processes in nature: selection („best survives" principle), sexual genetic crossover and random mutation (genotype alteration) rarely applied. The search is guided by the fitness function, which is the performance measure of the individuals. The individuals can have any form, from string to tree [6].

In short, a GA initially generates a population of feasible pseudo-random candidate solutions. In every step, called generation, the algorithm evaluates each individual in population, selects parents in the population, applies crossover to obtain new candidate solutions from selected parents and applies mutation to favor diversification in population. The newer individuals are added to the current population or, if a fixed dimension population is used, an accept-reject mechanism determines the content of the new population, which becomes current in the next generation. After many generations (hundreds or thousands), the best individual(s) in the last current population is/are returned as optimal solution(s).

By crossover, the "genetic material" of parents (in fact, the parents' structure and data) is combined to produce offspring which inherit characteristics of them. Mutation introduces new genetic material in population, thus completing the crossover function, which regularly cannot do that. The selection operator interferes in guiding the evolution by favouring survival of the fittest.

GAs are, among other methods, evolutionary search algorithms. They can be applied to performance-based search problems - problems where (1) adequate performance functions (fitness) can be defined so that members in the population can be fitness-ordered and (2) the set of solutions contains some possible solutions besides solutions that perfectly solve the problem [7].

B. ACO

Ant Colony Optimization, proposed by Marco Dorigo in 1992 [8], simulates the group behavior of ants which have the ability to find the shortest path to the food, by pheromone communication. The pheromone quantity accumulated on every covered route being proportional with covering frequency, it guides the colony to identifiy the optimal solution (the optimal route in the graph of routes identified by all the ants). In computer science, ACO is adequate to combinatorial optimization problems that try to identify optimal paths to goals.

To apply ACO, the graph of candidate components to be added to solutions has to be built. Initially, all the components have zero pheromone. At every iteration of ACO procedure, a number of agents called artificial ants, are launched in the graph of components and each of them builds a solution (a path in graph), step by step, starting from a random valid component. At every step, the pheromone for the visited node is updated and the quality of partial solution is updated. The pheromone quantity on the chosen component is proportional with the partial solution quality and inverse proportional with pheromone evaporation rate (if the procedure includes this mechanism to avoid the fast convergence to suboptimal regions of search space) [9, 10].

Interesting in ACO is the fact that agents moves independently and concurrently in the graph and every agent is complex enough to find a solution but it is ordinarily poor qualitative. The good solutions emerge only as a result of collective interractions between agents. This is a distributed learning process where individual agents are not adaptive, but they adaptively modify the way in which the process is perceived by other agents. On the final quality of a solution, the individual influence of an agent is not relevant, but the colony influence is a major one.

As we can see, the overall contrast between ACO and GA consists in the different way to generate candidate solutions.

The notable feature of ACO is using experience of agents in previous stages to guide the search through pheromone on candidate components.

C. PSO

Particle Swarm Optimization, proposed by Kennedy and Eberhart in 1995 [11], imitates the group behavior in birds flock, fish banks and other living beings, regarding the efficiency in attaining a destination.

The candidate solutions in PSO, named particles, are points or surfaces in metric multidimensional spaces, with a velocity and position associated. In PSO algorithm, the particles "fly" in the search space following the current optimal particles on basis of the adaptable speed which directs their moves, and on basis of a memory which retains the best visited location in the past by all agents.

The PSO advantages are the simplicity of implementation and the low number of parameters to adjust, but to note the difficulties of the method to avoid the local optima.

D. Scatter Search

SS, designed by Glover in 1986 [2], is also an evolutionary method, being distinct from other evolutionary computation methods by the manner of combining candidate solutions to create new ones and by the process of exploiting them. The SS

method, built on the principles regarding the surrogate constraint framework, is organized to capture information not present in the original candidate solutions, named vectors, and uses auxiliary heuristic methods both for selecting the elements to be combined and for generating new vectors [12].

The candidate solutions in SS are points or surfaces in Euclidean space, called vectors. The method uses a small population of vectors (generally below 20 individuals), called reference set, that evolves by a procedure which selects in a systematic way vectors to be combined in order to generate new vectors. Two, three or many vectors can be combined at a time to generate new combinations.

Initially, a diverse good solution vectors reference set is generated. Then, new solution vectors are generated as structured combinations of two or many vectors in the current reference set and a number of the best such solutions are chosen to be part of the next reference set after an heuristic improvement is made on them; this is repeated until the reference set is unchanged. To diversify the solution set the process is re-started from the first step and all these are repeated a fixed number of iterations.

The main characteristic of SS is that its structured combinations are designed to create weighted centers of selected subregions in the search space [12]. The method allows unfeasible vectors to be added to the reference set, with the goal of maintaining a diverse set. SS uses also an adaptive memory, by keeping the good solutions in a diverse collection of elite solutions. Both diversity of sets and quality of solutions participate in guiding the search towards favorable regions.

Path Relinking is a generalization of Scatter Search, where paths between and beyond selected solution vectors in neighborhood space are generated, instead of vector combinations in Euclidean space [13].

Comparing to GAs, that work with large populations to maintain a good enough diversity, SS systematically injects diversity to the reference set by combination mechanisms [12].

IV. Comparative Analysis

In this section, GAs, ACO, PSO and SS are compared on many criteria (see Table 1): the candidate solution form, the needed problem representation, the search guiding mechanism, general applying scope and other features.

As Table 1 shows, the analyzed evolutionary computation techniques (GA, ACO and PSO) are more similar in many aspects, in contrast with SS, which is not a stochastic metaheuristic.

TABLE I. GA, ACO, PSO, SS: Comparative Analysis

Comparing criteria	**Metaheuristic**			
	GA	***ACO***	***PSO***	***SS***
Form of candidate solution	Individual (representation of an artificial chromosome)	Trail / Artificial ant	Particle with speed and position	Solution vector
Candidate solutions are	Complete	Partial (zero or many componenents)	Complete	Complete
Problem representation	Genetic encoding for the individuals	Graph of candidate components	Metric encoding for the particles	Metric encoding for the vectors in Euclidean space
Search guided by	Genetic selection (to transmit genetic characters to the next generations) based on the fitness of individuals	Pheromone (historic quality) on the candidate components	Fitness of the particles	Diversity of reference set and quality of solution vectors, elite solutions set
Adequate for	Any type of combination of any type of values (e.g. string, tree)	Identifying paths to goals	Identifying paths to destinations, metric (real) multidimensional spaces	Metric (multidimensional) spaces
Other features	• Population initialization + update in generations • Genetic operators: selection, crossover, mutation	• Agents cooperate • The agent's experience influences the search	• Population nitialization + update in generations • A guided mutation • No use of selection and crossover operators	• The structured combinations are designed so that weighted centers of selected subregions are created
	• Big population dimension (≈ 100-10 000); • Stochastic methods; • Robust to loss of individual candidate solutions and to environmental changes; • Simulate natural intelligence emerged in populations of living beings; • The collective behavior in the self-organizing, descentralized system (the population) prove to be intelligent; the candidate solutions' intelligence is not good enough to optimally solve the problem; • Candidate solutions communicate, direct or indirect, with each other; • Achieve a parallel search in the search space; • Co-evolution of candidate solutions, mainly based on group experience; • Work with a population of simple agents (candidate solutions), which interact locally and with the environment;			• Small population dimension (5-20)

Comparing criteria	Metaheuristic			
	GA	***ACO***	***PSO***	***SS***
	• Need an adecquate setting of parameters' values; • Assign a quality measure to each candidate solution, that guides the search towards beneficial regions; • Return multiple (different) solutions.			

V. CONCLUSION

For the hard (optimization) problems, as most real world problems are, the traditional mathematical methods prove to be inadequate; the main reason for that is their time computational expensiveness, even infeasibility, in returning optimal solutions. Instead, metaheuristics, even if do not guarantee to find the optimal solution(s), they find near optimal solution(s) in a highly reduced computational time. For the most practical hard problems, this kind of solving is good enough.

Metaheuristics are young methods, designed after '70s. Most of them were proposed in the last two decades and many are presently developed.

The population-based metaheuristics are evolutionary algorithms, where evolution can be viewed as a travel in search space, on many ways, in many generations, towards regions with better and better performance. Such methods are Genetic Algorithms, Ant Colony Optimization, Particle Swarm Optimization, Scatter Search, Artificial Immune Systems, Bees Algorithm, Firefly Algorithm etc.

The achieved comparative analysis is a useful guide when try to apply one or many metaheuristic(s) to a problem, firstly regarding the problem representation and candidate solutions. The appropriateness for the problem type is another key aspect in choosing a metaheuristic. The results offer a global perspective over GA, ACO, PSO and Scatter Search, making easier to approach any of these on a specific problem, both for a beginner and for the advanced users.

REFERENCES

[1] K. Deb, "Multi-objective genetic algorithms: Problem difficulties and construction of test problems", Evol. Comp. J, vol. 7, pp. 205-230, 1999.

[2] F. Glover, "Future Paths for Integer Programming and Links to Artificial Intelligence", Comp. Oper. Res., vol. 13, pp. 533–549, 1986.

[3] S. Luke, Essentials of metaheuristics, Lulu, online version 1.1, 2011.

[4] E. S. Nicoară, GA-based Control of Multi-objective Flexible Job Shop Scheduling Processes (in Romanian), Ph.D. Dissertation, Informatics Dept., Petroleum-Gas University of Ploieşti, Ploiesti, Romania, 2011.

[5] D. B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, Piscataway, NJ: IEEE Press, 1995.

[6] J. H. Holland, Adaptation in Natural and Artificial Systems, MIT Press The University of Michigan Press, Ann Arbor, 1975.

[7] O. J. Sharpe, Towards a rational methodology for using evolutionary search algorithms, Ph.D. Dissertation, School of Cognitive and Computing Sciences, University of Sussex, 2000.

[8] M. Dorigo, Optimization, Learning and Natural Algorithms, Ph.D. Dissertation (in Italian), Politecnico di Milano, Italy, 1992.

[9] M. Dorigo and G. Di Caro, The ant colony optimization meta-heuristic, in New Ideas in Optimization, D. Corne, M. Dorigo, and F. Glover, Eds., McGraw-Hill, 1999.

[10] M. Dorigo and T. Stutzle, „The Ant Colony Optimization metaheuristic: algorithms, application and advances", Int. S. Oper. Res. Man. Sc., vol. 57, Handbook of Metaheuristics, F. Glover and G. Kochenberger, Eds., Kluwer Academic Publishers, Norwell, MA, 2002, pp. 251-285.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization", Proc. of IEEE Int. C. on Neural Networks, Perth, Australia, IEEE Service Center, Piscataway, NJ, 1995, pp. 1942-1948.

[12] F. Glover, M. Laguna, and R. Martí, "Fundamentals of Scatter Search and Path Relinking", Contr. Cyb., vol. 39, 2000, pp. 653-684.

[13] F. Glover, "Genetic Algorithms and Scatter Search: Unsuspected Potentials", Stat. Comp., vol. 4, 1994, pp. 131-140.