Swarm Intelligence Algorithms Parameter Tuning

Milan TUBA
Faculty of Computer Science
Megatrend University of Belgrade
Bulevar umetnosti 29, N. Belgrade
SERBIA
tuba@ieee.org

Abstract: - Nature inspired metaheuristic algorithms are recently successfully used to find suboptimal solutions to hard optimization problems. These algorithms mimic different nature phenomena in hope that nature's implicit intelligence will help to guide the search in untractable problems. Swarm intelligence algorithms are a class of nature inspired algorithms based on collective intelligence of colonies of ants, bees, fish etc. They have a number quantitative and qualitative parameters that can be adjusted. Such adjustments are not allowed for specific test problems but only for a whole class. When some adjustment works for number of classes it can be incorporated into the generic algorithm as a new qualitative parameter (optional modification). In this paper we describe how successful application of pheromone correction strategy for the ant colony optimization (ACO) algorithm on three different graph problems is incorporated in ACO software framework as a module.

Key-Words: - Swarm intelligence, Metaheuristic optimization**,** Ant colony optimization, Nature inspired metaheuristics

1 Introduction

Most real-life problems can be represented as some kind of optimization problem. Sometimes direct minimization or maximization of some numerical attribute is required, while in other situations first some qualitative property has to be numerically estimated and then optimized. Easy optimization problems were solved long time ago so nowadays only hard problems are of research interest. Many discrete (combinatorial) as well as some continuous optimization problems are intractable, but of great practical interest. Traveling salesman problem (TSP) is a classic example that was researched for the longest period of time and because of that is often used as a benchmark. There are also a number of engineering design problems with mixed continuous and discrete variable and nonlinear objective function and nonlinear constraints that are used as benchmarks for optimization algorithms.

Unconstrained optimization problems can be formulated as minimization or maximization of *D*-dimensional function:

$$Min\ (or\ max)\ f(x),\ x=(x_1,x_2,x_3,\ldots x_D) \qquad (1)$$

where *D* is the number of parameters to be optimized.

This research is supported by Ministry of Science, Republic of Serbia, Project No. 44006

General constrained optimization (CO) problem is to find *x* so as to:

minimize $f(x),\ x=(x_1,\ldots,x_n)\in R^n$ where $x\in F\subseteq S$

The objective function f is defined on the search space $S\subseteq R^n$ and the set $F\subseteq S$ defines the feasible region. The search space S is defined as an *n*-dimensional rectangle in R^n. The variable domains are limited by their lower and upper bounds:

$$l_i \le x_i \le u_i,\ 1\le i\le n$$

whereas the feasible region $F\subseteq S$ is defined by a set of *m* additional constraints ($m\ge 0$):

$$g_j(x)\le 0,\ \text{for}\ j=1,\ldots,q$$
$$h_j(x)\le 0,\ \text{for}\ j=q+1,..m$$

2 Hard Optimization Problems

The main problem with hard optimization problems is that there is enormous number of suboptimal solutions or local minima and there is no guidance how to search for global minimum. Standard down-hill methods in this situation fail. Typical example of such function that is used as a benchmark is Rastrigin function that is a sphere modified by small cosine waves.

The oldest way to deal with such problems is Monte-Carlo method. It is equivalent of trying to find the deepest point in the oceans by measuring many times the depth at random locations and hoping that best measurement will be close to the global optimum. While Monte-Carlo method is usable for some applications, its blind search is not sufficient for many others.

What may improve Monte-Carlo method's blind search is some heuristic that includes some way of exploitation of discovered good solutions. In such algorithms there is an attempt to implicitly understand the nature of the objective function.

Let us first consider the simplest benchmark function, a *n*-dimensional sphere which is continuous, convex and unimodal function. Global minimum value for this function is *0* and optimum solution is *x=(0, 0, . . . , 0)*. Surface plot is shown in Fig. 1.

Definition of *Sphere* function:

$$f(x)=\sum_{i=1}^{n} X_i^2 \qquad (1)$$

where *x* is in the interval of *[-100, 100]*

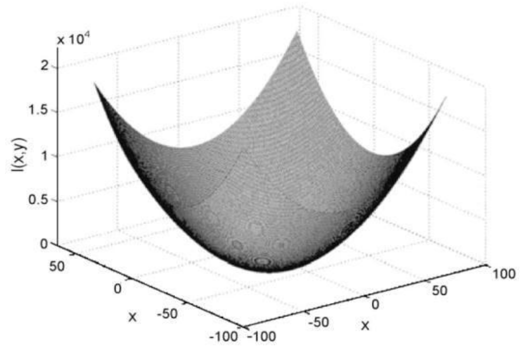

Fig. 1 Sphere function

Even for such a simple function minimum may not be approached close enough even after huge number of random guesses. If, however, any kind of exploitation of found good points is included, the search will very quickly converge to global minimum. The reason is very simple, even without knowledge of any derivatives or gradient, abandoning bad points and continuing search in the vicinity of good points will implicitly determine the down-hill direction. The only problem is to adequately define the vicinity: not to go too far to leave minimum region, and not to make the vicinity too narrow to slow the convergence.

The next level of complexity may be *Rastrigin* function which is based on *Sphere* function with the addition of cosine modulation to produce many local minima. Thus the function is multimodal. The global minimum value for this function is 0 and the corresponding global optimum solution is *x = (0,0,…,0)* .

Definition of *Rastrigin* function*:*

$$f(x) = 10n + \sum_{i=1}^{n}(X_i^2 - 10\cos(2\pi X_i)) \qquad (2)$$

where *x* is in the interval of *[-5.12, 5.12]*. Surface plot is shown in Fig. 5.

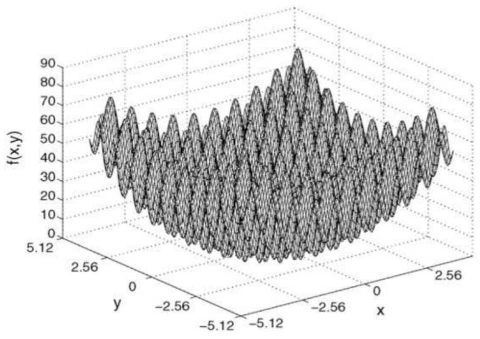

Fig. 2 Rastrigin function

Huge number of small waves that form local minima will make the previous intelligence fail since it will get trapped in local minima. One way to avoid it is to increase the vicinity to allow for exploration and leave one promising region in hope to find better one. The other way may be to consider some number of attempts in defined neighborhood as a cluster and represent it with the best element. If clusters are larger than small waves such group of representatives will filter out waves and show the structure of underlying sphere where previous down-hill intelligence apply.

Thus, in this case blind exploitation will trap the search in local minima and will not be able to leave them. If the waves are made larger it will be more difficult to leave local minima and more exploration will be necessary.

There are higher and higher levels of complexity where more and more elaborate intelligence would be necessary to grasp the function behavior. Extreme example on the other side of complexity is a function that has global minimum hidden in a very narrow deep pit that is located at unpromising high place. Any search will lead away from that position, the pit can be discovered only by random luck. In that case any exploitation will lead to failure, the best policy is complete exploration.

We can imagine that more and more complex examples can be introduced up to a point where any explicit understanding will stop. That is where interesting problems are and where we hope that hidden and implicit collective swarm intelligence will find a way to handle the situation.

3 Swarm Intelligence

As shown before, many interesting practical problems are intractable and no exact algorithm can find solution within reasonable time. In this rather hopeless situation researchers turned from mathematically exact methods to belief. The nature is doing miraculous things. We know the results, but we do not understand the mechanism. For hard optimization problems we try to mimic some nature processes. Older attempts included simulation of evolution (through genetic modifications and survival of the fittest) [1] and simulated annealing [2]. Recently, swarm intelligence become prominent using the fact that extremely simple individuals exhibit miraculous collective intelligence. Examples include ants colonies [3], honey bees colonies [4], flocks of birds, schools of fish etc. [5].

These nature inspired metaheuristics simulate various natural phenomena. We talk about bee colony food finding or ant colony path finding, but in essence, in all these diverse mimicking we do two things. We exploit good found solutions, but also go to unknown places in order to avoid being trapped in local minima. The successfulness of any such algorithm is determined by proper balance between exploitation and exploration. This balance is maintained by adjusting certain parameters and also by applying some rules in certain situations. Sometimes these are numerical parameters in appropriate range, in other situations they may be indicators that switch on and off certain algorithmic procedures. In any case, it not allowed to adjust parameters for each problem individually since it makes algorithm too specialized. Parameters can be adjusted for a whole class of problems. However, parameters can be changed dynamically, as a part of algorithm's intelligence. By doing such adjustments algorithm can become much better for some class of problems (off course, according to NFL theorem, it cannot become universally good for all problems).

This paper will demonstrate few successful examples of such adjustments.

4 Example: ACO pheromone correction strategy

The basic idea of ACO is to imitate the behavior of ants in a colony while gathering food. Each ant starts from the nest and walks toward food. It moves until it reaches an intersection, where it decides which path to take. In the beginning it seems as a random choice, but after some time the majority of ants are using the optimal path. This is possible because the colony works as a group and not just as individual ants, and the way it is achieved is by using pheromone as a collective memory for the ants in the colony. Each ant deposits pheromone while walking, which marks the route taken. The amount of pheromone indicates the usage of a certain route. Pheromone trail evaporates as time passes. Due to this, a shorter path will have more pheromone because it will have less time to evaporate before new pheromone is deposited ant that path will be chosen more often.

Presented behavior can be turned into a computational system [3] in the following way:

$$s = \begin{cases} \arg\max_{u \notin M_k} \left\{ {\tau_{rs}}^{\alpha} {\eta_{rs}}^{\beta} \right\} & , q \le q_0 \\ S & , q > q_0 \end{cases} \tag{1}$$

$$p_{rs}^k = \begin{cases} \dfrac{{\tau_{rs}}^{\alpha} {\eta_{rs}}^{\beta}}{\sum\limits_{u \notin M_k} {\tau_{ru}}^{\alpha} {\eta_{ru}}^{\beta}} & , s \notin M_k \\ 0 & , s \in M_k \end{cases} \tag{2}$$

Equations (1) and (2) describe the probabilistic decision method that an artificial ant *k*, currently at node *r*, after visiting nodes in M_k uses for choosing the next node *s*.

The pheromone trail is maintained using two types of updates. Global update is used to reward good paths, or in other words, more pheromone should be deposited on better paths, which is obtained by the following formula

$$\tau_{ij} = (1-e)\tau_{ij} + e\Delta\tau^k \quad , \forall ij \in B^k \tag{3}$$

The local updating is used to avoid creation of a very strong edge used by all ants, and it emulates pheromone evaporation. Every time an ant chooses an edge, it loses some pheromone by the following formula where τ_0 is a predefined constant.

$$\tau_{ij} = (1-e)\tau_{ij} + e\tau_0 \tag{4}$$

4.1 Pheromone Correction

The algorithm described in the previous section corresponds to the elitist ant system (EAS) variation of the ACO. In this version of the algorithm only the global best solution (or in some variations only the iteration best solution) deposits pheromone. It increases the efficiency of the basic ACO by intensifying the search near the global best solution, or in other words, making its search more greedy. One of the main problems of this version is the early stagnation of the algorithm. This is due to the fact that potential parts of the solution that are not near the global best solution loose more and more pheromone at each iteration because of the pheromone evaporation. Eventually it becomes practically impossible for these parts to be chosen by ants.

Min-max ant system (MMAS) [6] is an improvement of the EAS that tries to solve this problem. The improvement is done by adding an extra constraint that all pheromone values are bounded, $\tau_i \in [\tau_{\min}, \tau_{\max}]$.

Another approach to solve the stagnation problem is the minimum pheromone threshold strategy (MPTS) [7]. The MPTS uses a minimum threshold value τ_{mt} that is bounded $\tau_{\min} < \tau_{mt} < \tau_{\max}$. In the beginning of the algorithm it is set to some initial value and then adjusted during the search, depending on the performance. This adjustment is usually performed by dividing the τ_{mt} by a factor *k* at a predefined number of steps. Thus the MPTS avoids reinitialization of the pheromone trail and explores the solution search space more systematically. No loss of information occurs related to the pheromone trail reset, while the good properties of the MMAS are preserved.

4.2 Our Improvement

In this section we present a new hybridization of the ACO for avoiding stagnation in the local optima. The idea is to use the information about the best-found solution to perform some corrections on the pheromone trail. The concept of this hybridization is suspicion that some elements of the global best solution are not good. In our hybridization we direct to new search areas that are less suspicious which means with less undesirable properties. Directing the search is done by greatly decreasing the pheromone trail values at suspicious points.

A big drawback of MPTS is that added vertices are chosen just for having low values of τ_i which is a relatively random process. As a result, vertices that do not belong to good solutions are often reintroduced in the search. Once the pheromone value for a vertex is increased, it will take a long time for it to be removed from the intensively tested group. In our hybridization however, we do not add vertices to the "popular" group but rather remove vertices with suspicious properties, making the group smaller. In the following iterations ants will first select vertices from the popular set, and when none are left, vertices with better properties. This way we direct to new search areas that are less suspicious which means with less undesirable properties.

4.3 Application to the MWVCP

One of the classical graph theory problems is the minimum weight vertex cover problem (MWVCP). The problem is defined for an undirected graph $G = (V, E)$ where *V* is the set of vertexes, *E* is the set of edges and weights are assigned to each vertex in the graph. A vertex cover of graph *G* is a set of vertexes $V' \subseteq V$ that has the property that for every edge $e(v1,v2) \in E$ at least one of vertexes *v1* or *v2* is an element of *V'*. A minimum weight vertex cover is the vertex cover with the minimum sum of weights of the belonging vertexes. It has been shown that this problem is NP-complete even when it is restricted to a unit-weighted planar graph with the maximum vertex degree of three. A large number of real life problems could be converted to this form. An example could be the optimal positioning of garbage disposal facilities.

We implemented our concept of suspicious elements for the MWVCP [8]. When we observed the solutions generated after a small number of iterations of the ACO algorithm, we noticed that sometimes vertices with no effect were found in the solution. "No effect" is used here in the sense that if these vertices were removed from the solution set, the remaining part would still remain a cover set. Using this observation, we developed a criterion for calculating the suspicion that a vertex should not belong to the best solution:

$$Sus(i, V') = w(i) * NCE(V' \setminus \{i\}) \qquad (5)$$

The suspicion is formulated in Eq. 5 by function *Sus*. The function *NCE* gives the number of covered edges by a set of vertices for graph *G(V,E)*. An edge *e* is considered covered by the set *V'* if:

$$\exists (a \in V')(\exists b)((a, b) = e)) \qquad (6)$$

As it can be seen from Eq. 6, vertex *i* with a high weight and the set $V' \setminus \{i\}$ that covers a large number of edges is highly suspicious. The next step in our hybridization is to select a random number *RK* of vertices with the highest values of *Sus*. For each such vertex (i) the probability of being selected is:

$$p_i(selected) = \frac{RK - RankSus(i)}{RK} \qquad (7)$$

If these criteria are satisfied, we apply previously defined correction algorithm to the pheromone trail.

4.4 Application to the TSP

The Travelling Salesman Problem is the oldest and best researched NP-complete problem that is often used as a benchmark. We implemented our pheromone correction by analysis of the ACO algorithm for the TSP when it gets trapped in local optima where it indicated which corrections should be made, or more precisely what should not appear in the shortest path. There are two simple criteria that can be used on the edges belonging to the best found tour: very long edges and intersecting edges are very unlikely to be a part of the optimal path. The next step was to find a way to, without major corrections to the ACO algorithm, remove suspicious elements from the ants search path. The solution was to significantly lower the amount of pheromone on randomly selected highly suspicious edges belonging to the best path and letting the colony resume its search.

We have divided the correction into two parts: one considering the edge lengths, and the other one that is related to intersecting edges. All intersecting edges are considered highly suspicious and the pheromone trail is corrected on them. Edges for which pheromone trail correction will be applied due to their length is defined in the following way. First we define a heuristic for suspicion *Sus(rs) = length(rs)*. Using this heuristic we define the probability of edge *rs* being selected for pheromone correction

$$p_{selected}(rs) = \frac{RK - RankSusp(rs)}{2 * RK} \qquad (8)$$

The final step is to lower the pheromone trail for the selected edges:

$$(\forall (rs) \in Selected)(\tau_{rs} = \delta\tau_{rs}) \qquad (9)$$

where τ_{rs} is the amount of pheromone deposited on edge *rs* and δ is a fixed small constant that is used to significantly reduce the value of pheromone on edge *rs*.

This correction algorithm is applied if stagnation occurred and the criterion is that there was no change to the global best solution in at least n iterations.

4.5 Application to the MCDSP

A dominating set for a graph *G(V,E)* is a subset of vertexes $D \subseteq V$ that has a property that every vertex in *G* either belongs to *D* or is adjacent to a vertex in *D*. Finding the dominant set with the smallest possible cardinality among all dominating sets for a graph is one of the standard NP-complete problems. A very important variation of the minimum dominant set problem is its connected version. We call a dominant set connected if it has the property that any node $n \in D$ can reach any other node $m \in D$ by a path that stays entirely within *D*. That is, *D* induces a connected subgraph of *G*. The minimum connected dominant set is the one with the minimum number of vertexes. The minimum connected dominant set problem (MCDSP) is also NP-complete. The MCDS problem has gained popularity due to its close connection to the mobile ad hoc networks (MANETs) and sensor grids.

To improve the performance of ACO we implemented a pheromone correction strategy similar to the one used for minimum weight vertex cover problem (MWVCP) [8]. The idea of this approach is to change the pheromone matrix by analyzing some of the properties of the best found solution. More precisely, when the search for a better solution becomes stagnant we update the pheromone matrix. We do this by using a simple heuristic function that describes the desirability of a vertex in the solution. For example, a vertex that is part of the solution and does not cover any vertexes solely by itself is not very desirable. For an undesirable vertex in the solution we greatly decrease the value of the pheromone and as a consequence, that vertex is not often chosen as a part of the solution in the following steps of the algorithm.

First, let us define $\eta(v,V')$ as the number of vertexes that vertex v which is part of the best found solution V' solely covers.

$$Sus = \frac{1}{1+\eta(v,V')} \qquad (10)$$

In Eq. 10 we have defined *Sus* as the undesirability of a vertex in the solution. The next step is to select a random number *RK* of vertexes which solely cover the smallest number of vertexes. For each vertex i in the solution the probability of it being selected for pheromone correction is:

$$p_i(selected) = \frac{RK - RankSus(i,V')}{RK} \qquad (11)$$

Finally, we defined a stagnation criteria for recognizing if the search has been trapped in a local minimum as a situation when there has been no improvement in the solution in n iterations of the ant colony. We use separate values n_1 and n_2 for the two pheromone correction methods.

5 Software Framework Module

The described pheromone correction for undesirable points method is general enough to be included a as part of the ACO algorithm. We have developed a software framework for ACO [9] that we used for our research. A framework is a special kind of software library, that is similar to an application program interface (API) in the class of packages, that makes possible faster development of applications. However, while an API consists of a set of functions that user calls, a framework consists of a hierarchy of abstract classes. The user only defines suitable derived classes that implement the virtual functions of the abstract classes. Frameworks are characterized by using the inverse control mechanism for the communication with the user code: the functions of the framework call the user-defined functions and not the other way round. The framework thus provides full control structures for the invariant part of the algorithms and the user only supplies the problem-specific details. Such environment was suitable to implement our new pheromone correction strategy as a module that can be called from the framework. The framework incorporates possibility of hybridization in the directions of adding local searches to elevate the quality of paths found, or the possibility of pheromone trail correction when the algorithm has reached stagnation. This second option was used and more effective ACO version for the mentioned graph problems was defined.

6 Conclusion

We have investigated a new pheromone correction strategy based on avoidance of undesirable parts of the solution. It proved to be efficient on three different NP-complete graph problems: the minimum weight vertex cover problem, the travelling salesman problem and the minimum connected dominant set problem. As a strategy general enough, using possibilities of previously developed ACO software framework, it was successfully incorporated as another qualitative parameter. It may be extended to additional problems simply by adding modules to the framework.

References:

[1] J. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, MA, 1992

[2] Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing, *Science*, Vol. 220, No. 4598, 1983, pp. 671–680

[3] Dorigo M, Gambardella LM: Ant colonies for the traveling salesman problem. *BioSystems* Vol. 43 No.2 ,1997, pp.73-81

[4] Karaboga, D.: An idea based on honey bee swarm for numerical optimization, *Technical report-tr06*, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005

[5] Kennedy, J., Eberhart, R.C., Particle swarm optimization, *Proceedings of the 1995 IEEE International Conference on Neural Networks*, Piscataway, NJ: IEEE Service Center, 1995, pp. 1942–1948

[6] T. Stutzle, H. H. Hoos, Max-min ant system, *Future Gener. Comput. Syst.* 16 (9) (2000) 889–914

[7] K. Y. Wong, P. C. See, A new minimum pheromone threshold strategy (mpts) for maxmin ant system, *Applied Soft Computing*, 9 (3) (2009) 882–888

[8] Jovanovic R, Tuba M: An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem, *Applied Soft Computing* 11(8), 2011, 5360–5366

[9] R. Jovanovic, M. Tuba, D. Simian, An object-oriented framework with corresponding graphical user interface for developing ant colony optimization based algorithms, *Wseas Transactions on Computers* 7 (12) (2008) 1948–1957