Abstract—Ant colony optimization (ACO) algorithms have been successfully used for solving many optimization problems. This paper explains how optimization problems can be solved using ACO algorithms as well as the traditional and recent applications of ACO algorithms.

Keywords—Ant Colony Optimization, Static Problems, Dynamic Problems.

I. INTRODUCTION

The behavior of ants has inspired the development of artificial distributed problem solving systems. In terms of biological systems, each ant has its own agenda and follows very simple rules; more complex global-level patterns emerge solely through the ants’ interactions with each other and their environment (without supervision or central control). There are two types of interactions between ants in a colony: direct interactions (involves tactile, visual, or chemical contact) and indirect interactions (initiated by individuals that exhibit some behavior that modifies the environment which in turn stimulates a change in the behavior of other individuals). Indirect interactions can solve difficult problems although they may be simple [1].

Ants are able to find the shortest path between their nest and food sources because of the chemical substance (pheromone) that they deposit on their way. The pheromone evaporates over time so that the shortest paths will contain more pheromone and will subsequently attract greater numbers of ants. This is an example of an autocatalytic process that will continue until a trail from the ant colony to the food source is established. Figure 1, reproduced from [2, page: 19], shows the experimental apparatus and the typical result of an experiment with a double bridge with branches of different length. In (a), all ants choose a path on a 50% probability. In (b), the pheromone amount increases faster and makes shorter path more attractive to the other ants [2].

The ant colony (rather than individual ants) can be seen as an intelligent entity for its great level of self-organization and the complexity of the tasks it performs. Natural ant colony systems inspired many researchers in computer science to develop new solutions for optimization problems [2].

ACO algorithms simulate the foraging behavior of some ant species [3]. ACO algorithms use two factors for guiding the search process. These are: 1) the pheromone values (numerical values as a simulation for the pheromone that real ants deposit on their way to and from their nest), and 2) heuristic information. There are two types of heuristic information used by ACO algorithms; static heuristic information (that is computed at the initialization time and then remains unchanged throughout the whole algorithm’s run, such as the distances between cities in the traveling salesman problem (TSP)) and dynamic heuristic information (that depends on the partial solution constructed so far and therefore it is computed at each step of an ant’s walk). The main role of the heuristic information is to avoid constructing tours of very bad quality by biasing ants so that they build reasonably good tours from the very beginning. If no obvious heuristic information exists, using an ACO algorithm incorporating local search may be enough to achieve good results [4].

ACO algorithms can be defined either from the operations research (OR) perspective or artificial intelligence (AI) perspective:

- From the artificial intelligence (AI) perspective:

   It is one of the most successful strands of swarm intelligence (SI), the AI discipline whose goal is to design intelligent multi-agent systems by taking inspiration from the collective behavior of animal societies such as ant colonies, flocks of birds, or fish schools.

   Examples of “swarm intelligent” algorithms other than ACO are:

   - algorithms developed for clustering,
ant colony optimization algorithms belong to the class of metaheuristics. Before the term metaheuristics was widely used, metaheuristics were often called modern heuristics. A metaheuristic is a set of algorithmic concepts that can be used to define heuristic methods applicable to a wide set of different problems. In other words, a metaheuristic can be seen as a general-purpose heuristic method designed to guide an underlying problem-specific heuristic toward promising regions of the search space containing high-quality solutions. Therefore, a metaheuristic is a general algorithmic framework which can be applied to different optimization problems with relatively few modifications to make them adapted to specific problem.

Examples of “metaheuristics” algorithms other than ACO algorithms are: iterated local search, simulated annealing, and tabu search [4]-[5]. Figure 3 illustrates this definition.

II. APPLICATIONS OF ANT COLONY OPTIMIZATION

Ant colony optimization was introduced by means of the proof-of-concepts to the traveling salesman problem (TSP). Since then, ACO algorithms have been applied to many optimization problems. First, classical problems (other than the TSP) were tackled such as routing, assignment, scheduling, or subset problems. More recent applications include for example bioinformatics, multi-objective, and dynamic problems [5].

The early applications of ACO algorithms have been mainly concerned with solving ordering problems (where the order between the components of the solutions is important and the next component to be added to a partial solution is affected by the last added component) such as shop job scheduling [4]. One of the recent trends in ACO is to solve industrial problems proving that it is useful for real-world applications [7]-[8].

An artificial ant in ant colony optimization is a stochastic constructive procedure that incrementally builds a solution by adding solution components to a partial solution under construction. Hence, ant colony optimization metaheuristic can be applied to any combinatorial optimization problem for which a constructive heuristic can be defined. The real issue is how to map the considered problem to a representation that can be used by the artificial ants to build solutions.

An ACO algorithm can be consisted of three procedures:

- **Construct solutions:** manages a colony of ants that concurrently visit adjacent states of the considered problem by applying a stochastic local decision policy that uses pheromone trails and heuristic information. Once an ant has built a solution (or while building a solution), the ant evaluates the solution (or the partial solution) that will be used by the second procedure (pheromone update) to decide how much pheromone will be deposited.

- **Pheromone update:** is the process by which the pheromone trails are modified. The deposit of new pheromone increases the probability that those solutions components that were either used by many ants or that were used by at least one ant and which produced a very good solution will be used again by future ants.

- **Applying local search (optional):** is used to implement centralized actions which can not be performed by single ants. It can observe the path found by each ant in the colony and select one or a few ants (those that built the best solutions in the algorithm iteration) which are then allowed to deposit additional pheromone on the solution components they used. Once ants have completed their solution construction, the solutions can be taken to their local optimum by applying a local search routine. Such a coupling of solution construction with local search is a promising approach. Because ACO’s solution construction uses a different neighborhood than local search, the probability that local search improves a solution constructed by an ant is quite high. On the other hand, local search alone suffers from the problem of finding good starting solutions (provided by the artificial ants) [4]. There exist problems for which local search is of limited effectiveness such as very strongly constraint problems. These are problems for which local search efficient polynomial neighborhoods contain few solutions or none at all and local search is of very limited use [6].
ACO can be summarized by the following:

- There is a set of constraints defined for the given problem,
- There is a set of components,
- The states of the problem are defined in terms of sequences of finite length over the elements of the solution components,
- A solution is an element of the set of all feasible states verifying all the problem requirements, and
- There is a cost associated to each candidate solution.

All the previous characteristics can be represented in the form of weighted graph (called construction graph) \( G = (N, A) \) where \( A \) is the set of edges that connects the set of nodes \( N \) (components). Hence, we have:

- The components are the nodes of the graph,
- The states correspond to paths in the graph,
- The edges of the graph are connections/transitions, and
- The components and connections may have associated pheromone trails and heuristic values [4], [9].

Some guidelines of how to solve optimization problems by ACO can be summarized by the following:

- An appropriate problem representation which allows ants to build solutions by using the probabilistic transition rule. The main idea generally is to model the problem as the search for the best path through a graph.
- Appropriate definition of the meaning of the pheromone trails i.e., the type of decision they bias. This is a crucial step in the implementation of an ACO algorithm.
- Appropriate definition of the heuristic preference to each decision that an ant has to take while constructing a solution i.e., define the heuristic information (problem specific) associated to each component or transition. The heuristic information is crucial for good performance if local search algorithms are not available or can not be applied.
- Appropriate definition of the probabilistic transition rule that determines which node an ant should visit next. It depends on the pheromone and/or the heuristic value.
- If possible, implement an efficient local search algorithm for the given problem.
- The pheromone update rule that specifies how to modify the pheromone trails that is an essential part of the probabilistic transition rule.

Each ant in the colony has the following properties:

- It exploits the construction graph to search for the minimum cost feasible solutions for the problem being solved.
- It has a memory that it can use to store information about the path it followed so far. The memory can be used to: build feasible solutions, compute the heuristic values, evaluate the found solution, and retrace the path backward.
- It has a start state and one or more termination conditions.

When it is in a state:

- If no termination condition is satisfied, it moves to a node in its neighborhood of the current node.
- If at least one of the termination conditions is satisfied, then the ant stops.
- It selects a move by applying a probabilistic decision rule that is a function of: the locally available pheromone trails and heuristic values, the ant’s private memory storing its current state, and the problem constraints.
- When adding a component to the current state, it can update the pheromone trail associated with that component or with the corresponding connection.
- Once it has built a solution, it can retrace the same path backward and update the pheromone trails of the used components.
- It starts in the start state and moves towards feasible states building its associated solution incrementally [4], [9].

It should be noted that good-quality solutions can only emerge as the result of the collective interaction among the ants. This is obtained via indirect communication mediated by the information ants read or write in the variables storing pheromone trail values. This is a distributed learning process in which the single ants are not adaptive themselves but adaptively modify the way the problem is represented and perceived by other ants [4].

Some of these optimization problems are summarized in Table 1.

### Table 1: Examples of Optimization Problems

<table>
<thead>
<tr>
<th>Static Problems</th>
<th>Routing Problems</th>
<th>The Traveling Salesman</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment Problems</td>
<td>The Vehicle Routing</td>
<td></td>
</tr>
<tr>
<td>The Quadratic Assignment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Generalized Assignment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Single Machine Total Tardiness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Resource-Constrained Project Scheduling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Multiple Knapsack</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Set Covering</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Set Partitioning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Discovery of Classification Rules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The dynamic TSP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The routing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The k-objective symmetric TSP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The multicast routing</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A. Static Problems

(Are relatively straightforward) are those in which the characteristics of the problem are given once when the problem is defined and do not change while the problem is being solved. Examples are: routing problems (such as the traveling salesman problem (TSP) that plays an important role in the ACO research as many of the ACO algorithms were first tested on it for several reasons: it is an important NP-hard optimization problem that arises in several applications, it is a problem to which ACO algorithms are easily applied, and it is a standard test bed for new algorithmic ideas [4], the vehicle routing problem (VRP) [4], [12], and the sequential ordering problem (SOP) [13]), assignment problems - a set of items has to be assigned to a given number of resources [7] (such as the quadratic assignment problem (QAP) [12]-[13], the graph coloring problem (GCP) [14]-[15], and the generalized assignment problem (GAP) [4]), scheduling problems - the allocation of scarce resources to tasks over time [7] such as the job-shop scheduling problem (JSP) [13], the single machine total tardiness problem (SMTWTP) [16]-17, and the resource-constrained project scheduling problem (RCPSP) [18]), subset problems - the selection of a subset of items from some larger set subject to certain constraints (such as the multiple knapsack problem (MKP) [4], [19], the set covering problem (SCP) [20], and the set partitioning problem (SPP) [6], [17]), machine learning problems (such as the discovery of classification rules problem (DCRP) [21]), and bioinformatics problems (the application of computer technology to the management of biological information [22]) (such as application to protein folding that is one of the most challenging problems in computational biology [5], [23]).

B. The Multi-Objective Problems

Almost all the applications of ACO tackle problems for which solutions are evaluated according to only one objective. However, many real-world problems involve multiple often conflicting objectives and it is therefore highly desirable to extend the known best techniques to tackle such problems. These multi-objective combinatorial optimization problems replace the scalar value in single objective problems by an objective vector where each component of the vector measures the quality of a candidate solution for one objective [24], i.e., several objectives have to be simultaneously optimized. Hence, there is not usually a single best solution solving the problem but a set of solutions that are superior to the remainder when all the objectives are considered. These solutions are known as Pareto-optimal or non-dominated solutions (are equally acceptable as they satisfy all the objectives) while the remainder are known as dominated solutions.

An example is k-objective symmetric TSP where k=two or more different single objective and the TSP instance has the same number of towns [25]. Another example is the multicast routing problem (the construction of a multicast tree in a computer network to route a given traffic demand from a source to one or more destinations) [26].

C. The Dynamic Problems

The problem changes at run time and the algorithms must be capable of adapting online to the changing environment [13]. An example is the dynamic TSP in which cities can be added or removed at run time and the goal is to find (as quickly as possible) the new shortest tour after each transition [4].

Another example is the routing problem. Routing is one of the most critical components of network control. The generic routing problem in communications networks can be informally stated as the problem of building and using routing tables (the routing table of a generic node i is a data structure that says to data packets, entering node i, which should be the next node to move to among the set of neighbors of i) to direct data traffic so that some measure of network performance (is a function of the type of network and of the provided services) is maximized [12]-[13].

Let $G=(N, A)$ be a directed weighted graph, where each node in the set N represents a network node (with processing/queuing and forwarding capabilities) and each oriented arc in A is a transmission system (link) (with an associated weight defined by its physical properties).

Network applications generate data flows from source to destination nodes. For each node in the network, the local routing component uses the local routing table to choose the best outgoing link to direct incoming data towards their destination nodes.

ACO implementations for communications networks are those for connection-oriented networks routing and those for connection-less networks routing. In connection-oriented networks, all the packets of a same session follow a connection path selected by a preliminary setup phase. On the contrary, in connection-less networks, data packets of a same session can follow different paths. In both types of networks, best-effort routing (routing without any explicit network resource reservation) can be delivered. Moreover, in connection-oriented networks, an explicit reservation of the resources can be done. In this way, services requiring specific characteristics (e.g., bandwidth, delay…) can be delivered [13].

III. CONCLUSION

This paper explained the basics of ACO and how optimization problems can be solved using ACO algorithms. It explained the traditional and recent applications of ACO algorithms as well.

REFERENCES


