7.4 Ant Colony Optimization

Ming Lu

2022-04-02

The ant colony optimization (ACO) heuristic belongs to the class of swarm intelligence based methods where a collection of individuals, having a common goal, exchange information about their local actions to build collectively a global procedure that would lead everyone of the participants to the goal. Originally proposed by M. Dorigo in his Doctoral Dissertation Dissertation (Dorigo 1992), and Colorni et al. (1992a, b), as an optimization heuristic in Computer Sciences, the ant colony optimization paradigm has been since then successfully applied to many optimization problems, ranging from purely combinatorial, to problems in telecommunication networks, scheduling, machine learning, bioinformatics and finance (Dorigo and Stützle 2004; Stützle et al. 2011). We give a brief account of this heuristic and show an application to option valuation in the following sections.

# 7.4.1 The Basics of Ant Colony Optimization

The inspiration for the ACO heuristic comes from the way a colony of ants search for food and traces their way back to the nest. In this search various ants go out of the colony searching for food, and once an ant finds food, it takes a portion back to the nest leaving traces of pheromone on the ground, a chemical trail to guide other ants to the food source. The first ants to arrive with food to the nest have found shortest paths to the food sources, and the quantity of pheromone deposited in each trail is an indication of the quantity and quality of the food. All this information is gathered by other ants from the nest, which quickly go out for food through the currently best routes (those with higher pheromone traces and made by the first arriving comrades). This immediate reaction updates the importance of some trails, since previously laid pheromone slowly evaporates. As the food at certain sources reduces in quantity or quality the trail slowly disappear, as the intensity of pheromone diminishes, and ants start searching for other food sources.

It is clear that this is an appropriate heuristic for solving optimization problems which can be reduced to finding routes in graphs. Hence, the crux in the design of an ACO solution for a given optimization problem is to devise a constructive heuristic for building solutions (elements of ), as sequences of elements from a finite set of solution components, which one must determined. We may view as a map from sequences of elements of onto . Any solution is then seen as a path in a totally connected graph formed by the elements of , and a solution component is added to the partial solution by the constructive heuristic , which makes sure the constrains are met.

Example 7.4 To illustrate the crucial step just described for the design of an solution, consider the asymmetric traveling salesman problem (ATSP), where an instance is a completely connected directed graph with a positive weight associated to each . The vertices of the graph (elements of ) represent cities and the arc weights represent distances between the cities. The goal in this problem is to find the minimum weighted directed Hamiltonian cycle, that is, a Hamiltonian cycle in for which the sum of the weights of its arcs is minimal. The ATSP is an NP-hard problem (Garey and Johnson 1979) Then to find approximate solutions to this problem with the heuristic, consider as solution components the set of arcs, so , and the constructive heuristic consist on assembling arcs together in such a way that a path is built verifying the Hamiltonian restriction.

The basic ACO scheme is presented in Algorithm 7.6. The input is an instance of an optimization problem , and it is assumed that we have a constructive heuristic with the characteristics described above, which builds candidate solutions as paths formed with solution components from a finite set . Additionally, for a true ACO solution, in this constructive process for building solutions as paths, ants choose next solution component randomly with probabilities subjected to the pheromone model. These probabilities of moving from one solution component to another (or transition probabilities), are defined for the th ant, acting at time , by the equation

where is the pheromone value present in component at time is the heuristic information value of the solution component at time is the set of solution components that ant has not yet visited, and the parameters and , and , measure the relative importance of pheromone value and heuristic information.

This process of building a path by an ant, using the constructive heuristic to build solutions from the set of solution components, subjected to the constraints ,

and selecting a next component in the path using the transition probabilities given by Eq. (7.8), is encoded in the algorithm (line 5) as function BuildPath( . This function returns a path , built by the th ant. The quality of each path is evaluated by the objective function and the best path (or best solution) is updated accordingly (line 6). Then the path is saved in the set , which contains already constructed paths verifying the constraints up to time (or iteration) (line 7 ).

The pheromone values are initially set to some positive constant , and uniformly for all solution components. Afterwards, at each iteration , when all ants have built their paths, pheromone values are updated for all components that belong to a path in the set . This is what the subroutine UpdatePheromone does (line 9 ). The pheromone value of component is updated at time , according to the following rule

where measures the pheromone decay (or evaporation rate), and

with a path (candidate solution) containing component some positive constant, and is the total number of ants. Thus, Eq. (7.9) gives an update of the value of the pheromone present at time in solution component , as a convex combination of current value (which if it hasn’t been visited is the initial constant ) and the cumulative sum of the pheromone deposited in by all ants. In this manner, with the parameter , one can control the velocity of convergence of the algorithm towards local optima, by setting, for example, a high evaporation rate. A stopping criterion for algorithm is usually set as an upper bound in the number of iterations. When this bound is reached the best path found so far is given as output. The next section describes an application of the ACO heuristic to the valuation of American put options.

# 7.4.2 Valuing Options with Ant Colony Optimization

Keber and Schuster (2002) give an ant colony based heuristic to learn analytical expressions for the valuation of American put options on non dividend paying stocks. Their main idea is to define a context-free grammar to generate analytical expressions that are to be interpreted as functional approximations for

the valuation of stock options. Then an heuristic is used as search engine for producing the derivations of the grammar.

Let us begin fixing the notation. In a context-free grammar , the set is the vocabulary of the grammar and its elements are the terminal symbols, is the set of non-terminal symbols, is the start symbol, which is a member of , and is the set of production rules. All the sets and are finite. The language produced by the grammar , consists of all finite concatenations (strings) of elements of , which can be derived from the start symbol by successive application of the production rules in acting on the non-terminal symbols to produce elements in or . Formally,

where is the set of all finite strings over the vocabulary , and means that is derived from by finitely many successive applications of production rules from . The following example illustrates how a context-free grammar could be defined to obtain analytical expressions like the Black-Scholes option pricing formulas (Eqs. and 5.22).

Example 7.5 We require a grammar where the set of terminals should contain the basic items found in a real valued analytical equation (e.g. mathematical operators, real numbers, and some predefined variables). The following should fit our purpose:

The first five symbols represent, respectively, initial stock price (i.e. at time ), option’s exercise price, annual continuous risk-free interest rate, annualized time to expiration, and annual volatility. We have the ten first natural digits that are used to construct real numbers in a manner that will be explained shortly. We have symbols for the basic arithmetic operations, and some analytical functions, including the standard normal distribution .

The set of non-terminals contain symbols that can be expanded to terminals or other non-terminals, and the start symbol is a member of . Let

The production rules in describe how the non-terminals are mapped to other nonterminals or terminals to create analytical expressions. Consider the following rules:

The ” ” (or) separates the different alternatives for a rule acting on a non-terminal. A derivation consist in a sequence of replacements of a non-terminal symbol by another symbol, until a string containing only terminal symbols is obtained. Each step in the derivation conforms to a mapping from a non-terminal to another symbol defined by one of the production rules. Special attention deserves the set of rules for real , int and digit . These conform an efficient method to create numerical constants, integer or real, of any size, and which has been termed digit concatenation ( ’Neill et al. 2003). For example, the real number can be derived using these set of rules, as follows:

A derivation in the language of the grammar starts with the symbol , and results in an analytical expression on the variables , and . Let us see how the expression could be derived in this grammar.

We have abbreviated some steps of this derivation: the first replaces the derivation of the number done before; the other ellipsis are shortcuts for simultaneous applications of two or more obvious production rules.

Having defined the appropriate grammar, the next step is to use the ACO heuristic as search engine in the space of words . The adaptation of the heuristic is somewhat natural: the strings are the feasible solutions (the potential analytical expressions for valuing options), the solution components are the nonterminal and terminal symbols, paths are derivations in the grammar, ants begin

path construction process BuildPath is comprised of the production rules of the grammar plus the transition probabilities given by Eq. (7.8) to decide the alternatives of the rules to use in the next derivation.

The fitness function is defined as follows. Each path built by an ant, or derivation (an analytical expression), is a function on the variables contained in the terminals , so that on input data , subject to certain domain restrictions, delivers an output data . Therefore, to test the fitness of , several input data records are produced respecting the restrictions, for , and each corresponding output record is compared to a target output . The fitness function is then defined by , where is the aggregated deviations given by some error function (e.g.  could be a sum of squared errors).

Additionally, to the ACO heuristic (Algorithm 7.6) applied to the language ) and fitness function as described above, Keber and Schuster apply a reduction in size of the set in the UpdatePheromone routine (line 9 ), by deleting those paths (instances of analytic expressions) where pheromone has evaporated. The motivation for this adjustment is that the number of expressions in can grow exponentially, and hence some form of pruning the space search is needed.

For their experimental set up, Keber and Schuster used as training data a randomly generated sample of 1,000 American put options on non-dividend paying stocks, described as tuples , where is the “exact” value of an American put option on non-dividend paying stock at , computed by the finite difference method, and , where represents the killing price (computed by some known numerical procedure). These sample tuples were transformed into an input data record and a corresponding target output record , for , and for the aggregated deviations function they used the sum of squared errors . The domains of values for the variables were restricted as follows: , and and . The number of ants were taken as , and for the relative importance of pheromone and heuristic information they used and ; was set to , and the stopping criterion is to halt after 100,000 iterations.

Keber and Schuster found in their experiments as one of their best analytical approximations derived by their ACO-engined grammar, the following expression

where

A formula that has an amazing similarity in its structure to the Black-Scholes formula for valuing European put options. Validation of expressions like these, and other best solutions obtained by their program, against 50,000 randomly generated American put options, resulted in a 96% of accuracy for some cases, outperforming some of the most frequently used analytical approximations for American put options.