[Year]

Formative Overview

ANT COLONY OPTIMISATION: A UNIFIED FRAMEWORK

LUKE MCCANN

Introduction

In recent years artificial intelligence (AI) and machine learning (ML) have made a comeback, after an initial dark period, in the industry of research and academia regarding computational systems. As such, many researchers have taken an interest in accelerating the growth of specific technologies in these areas including; meta-heuristics, neural networks (NN's), and evolutionary computing (ECO). Autonomous systems offer a great advantage to many industries, allowing for machines to explore zones or perform tasks which hold a danger to human health, or simply with greater levels of accuracy.

Swarm intelligence (SI) is defined by the emerging complex self-organising behaviours of groups of autonomous agents whom as a single agent sport low level intelligence (*Louis P. Walters, 2011)*. While much of the research in SI is highly theoretical there are a variety of nature-inspired algorithms which have been proposed.

Meta-heuristics, high level procedures designed to search for an adequate solution to some optimisation problem with incomplete datasets, aim to implement some of these algorithms. While globally an optimal solution is not guaranteed to be discovered, many of these algorithms implement stochastic optimisation. Some of the most successful include;

- Ant colony optimisation (ACO)
- Particle swarm optimisation (PSO)

Project

In this project I aim to implement a version of the Ant Colony Optimisation (ACO) algorithm as a framework. The targeted audience being future developers of who wish to implement the algorithm to problems. The framework should be generic enough yet allow the user to implement the algorithm with ease, as such research has been conducted into not only ACO but the building of a framework as well.

As of this moment experimentation has begun. This includes building ACO simulations for varying scenarios; Shortest Path First, Network Routing Simulations etc.. from these experiments the optimal build is decided to be the Multi Objective Ant Colony Optimisation algorithm (MOACO).

MOACO prevents the algorithm becoming stuck in local optima via the implementation of an evaporation method ensuring that agents do not get stuck in their own positive feedback loop. Some randomness is required in this algorithm as to encourage exploration of potentially better solutions. The algorithm itself is a branch of genetic algorithm, solutions are evolved over time and can adapt to issues. For instance, MOACO implemented in a network is able to adapt should one node become unavailable and re-calculate the best possible routing solution.

The next step of this project is to begin the implementation of a framework. This will be a package of functions which other developers may download and implement into their own projects. This framework should be generic enough to allow users to apply the solution to any problem of a non-deterministic polynomial-time hardness.

Constructing Solutions

When constructing solutions, the ants follow the strongest pheromone trail. To allow the ants to deliberate and discover solutions of improved efficiency over the current best it is fundamental to the algorithm that some form of randomness be applied. The following equation denotes the probability of selecting a single component to the solution;

$$\rho_{ij}^{k} = \frac{[t_{ij}]^{\alpha} * [n_{ij}]^{\beta}}{\sum l \in N_l^k [t_{il}]^{\alpha} * [n_{il}]^{\beta}}$$

Figure 1. ACO Solution Equation
(*Shekhawat, Poddar, Boswal 2009*)

The amount of pheromone is represented via *T* with *i* and *j* being the nodes between which the ant will transpose. The parameter of which regulates the influence of the pheromone (T_{ij}) is denoted by the Greek letter alpha (α) while beta (β) is representative of the heuristic value in control of the edge desirability's influence (n) (*Shekawat, 2009*). The overall equation calculates the probability of which a single component within the solutions is selected.

Local Pheromone Update

Each time a successful solution is constructed the pheromone update function runs. This acts as the imitation of the way biological ants lay pheromones (*Dorigo, 1992*). In the artificial model, pheromone updates are replicated by varying the level of deposited pheromone on a path dependent on the overall score of the completed solution. The amount of pheromone which is deposited ensues as a result of the following equation;

$$T_{ij} = (X - \rho)T_{ij} + \Delta T_{ij}$$

Figure 2. Pheromone Update Equation
(*Shekhawat, Poddar, Boswal 2009*)

The above equation is a superior method of implementation compared to the original AS system. This is due to the inclusion of pheromone evaporation conveyed as (ρ) allowing for inefficient solutions to be "forgotten" over time (*Bundgaard, Damgaard, Dacara & Winther, 2002*). The pheromone on any particular edge is denoted as $T_{ij,}$ when multiplied with delta this gives the quantity of deposited pheromone which in turn can be determined by;

$$\Delta T_{ij}^{k} = \begin{cases} X/C_k & \textit{If ij was used by agent} \\ 0 & \textit{Else} \end{cases}$$

$$T_{ij} \leftarrow T_{ij} + \sum_{k=1}^{m} \Delta T_{ij}^{k}$$

Figure 3. Pheromone Update Equation
(*Shekhawat, Poddar, Boswal 2009*)

Where (C^k) defines the total cost of constructing the solution. Typically, the parameter X is set to 1, as the parameter to adjust the deposited pheromone, however in some problem statements (such as TSP) this would be adjusted via a value such as the length of the agent's journey.