UNIVERSITÉ LIBRE DE BRUXELLES
Faculté des Sciences
Département d'Informatique

Development of an automatically configurable ant colony optimization framework

Aldar Saranov

Promoter : Prof. Thomas Stützle

Master Thesis in Computer Sciences

Academic year 2016 - 2017+1

Dedicated to my mother Lena, who was always sincerely supporting me

"*If one does not know to which port one is sailing, no wind is favorable.*"

Lucius Annaeus Seneca, 1st century AD

"*The general, unable to control his irritation, will launch his men to the assault like swarming ants, with the result that one-third of his men are slain, while the town still remains untaken. Such are the disastrous effects of a siege.*"

Sun Tzu, "The Art of War", 5th century BC

Acknowledgment

I want to thank my promoter, prof. Thomas Stützle, whose determination and competencies were leading me to the goal of the paper, and my friend, Alain, who made this precious time of my education possible.

Contents

1 Introduction 4-5 **1**

2 Background 20 **2**

2.1 Combinatorial Optimization Problems and Constructive Heuristics 5 . . 2

2.2 Ant Colony Optimization 10 3

2.2.1 Choice of pheromone trails and heuristic information 4

2.3 Iterated F-Race 5 4

3 Vehicle Routing Problem 5 **5**

4 Implementation 20-25 **6**

5 Experimental 20-25 **7**

6 Conclusion 4-5 **8**

6.1 Literature 8

Chapter 1

Introduction 4-5

Some species show an extreme degree of social organization. Such species (e.g. ants) have pheromone production and detection body parts and therefore seize an ability to communicate between each other in an indirect way. This concept has inspired the development of algorithms, which are based on the social behavior of the ant colonies called ant colony optimization algorithms. These algorithms allow to solve NP-hard problems in a very efficient manner. These algorithms are considered to be metaheuristics. The development of an ACO framework is the next step of formalizing this area. Such a framework can then be used as a tool to help resolving various optimization problems. This report gives a brief overview of the current state of the ACO research area, existing framework description and some tools which can be used for the automatic configuration of the framework.

In the chapter 2, we describe the theoretical foundations of the framework, that is required to develop. However unreasonable selection of a running configuration may lead to producing of low-quality results even after a long runtime execution. A simple decision is to perform certain configuration tuning before the actual executing. This chapter also introduces an efficient configuration software, that allows to obtain high-performing configurations for further framework exploiting.

The development of such framework would also require its application to some particular problem for testing and analysis purposes. Vehicle Routing Problem was chosen as the one. It is a widely researched NP-hard combinatorial optimization problem, therefore, it will allow us to perform benchmark comparison of the public available problem instances and its solutions with the solutions, that we obtain. In the chapter 3, we introduce formulation of Vehicle Routing Problem and also mention several variations thereof.

In the chapter 4, we describe thorough implementation details and decisions for the ACO framework, the way we adapt it specifically to VRP-problems and also show line-by-line description of the parameter space defined for the framework. We also explain there how do we attain high flexibility of the framework in order to adapt it for various problem types besides VRP.

In the chapter 5, one can see the experimental set-up, tuning results and interpretations that are made.

Chapter 2

Background 20

2.1 Combinatorial Optimization Problems and Constructive Heuristics 5

Combinatorial optimization problems (COPs) are a large class of mathematical optimization problems. These problems can be described as a grouping, ordering, assignment or any other operations over a set of discrete objects. In practice, one may need to resolve COPs, which have a large number of extra constraints for the solutions to consider them as admissible. Many of these problems which are still being thoroughly researched at the moment, belong to NP-hard discrete optimization problems. The best performing algorithms known today to solve such problems have a worst-case run-time larger than polynomial (e.g. exponential).

An Optimization Problem is a tuple [**?**] (Φ, ω, f), where

- Φ is a search space consisting of all possible assignments of discrete variables x_i, with $i = 1, ..., n$
- ω is a set of constraints on the decision variables
- $f : \Phi \to R$ is an objective function, which has to be optimized

The problem formulation describes an abstract task (e.g. find the minimum spanning tree of some graph), while the instance of a problem describes a specific practical problem (e.g. find the minimum spanning tree of a given graph G). The objective function is the sum of the weights of the selected edges.

[TSP and QAP description here, or not needed???]

2.2 Ant Colony Optimization 10

Since solving of such NP-hard problems by trying to find provably optimal solutions is unreasonable, one can apply heuristic algorithms, which more or less provide solutions with relatively good quality consuming reasonable quantity of resources (time/power, memory etc.).

An important class of such heuristic algorithms are constructive heuristics. Constructive heuristics start with an empty or partially built solution, which is then being completed by iterative extension until a full solution is completed. Each of the iterations adds one or several solution components to the solution. For example, greedy constructive heuristics add the best-ranked components by their heuristic values.

ACO also uses the notion of a heuristic values

Generally, heuristic values are assigned as constants at the start of the solution construction. However, in extensions one can use heuristic values, which are a function of the generated partial solution. These are called *adaptive* heuristics and normally they consume larger computer resources although they often lead to a better quality of the generated solutions.

[selectors description]

[rip-off the previous year paper, describe only those components that are used]

ACO algorithms are a class of constructive heuristics. Meta-heuristic is a top-level heuristic, which is used to improve the performance of an underlying, basic heuristic. ACO is such a metaheuristic that can be used to improve the performance of a construction heuristic. Features of ACO algorithms are the following.

- ACO algorithms are population-based algorithms. Solutions are being generated at each iteration.
- Solutions are being generated according to a probability-based mechanism, which is biased by artificial pheromones that are assigned to problem specific solution components.
- The quality of the generated solutions affect the pheromones, which are updated during the run of the algorithm.

Listing 2.1: General ACO pseudo-code

```
1  procedure ACO–Metaheuristic
2  repeat
3    foreach ant do
4      repeat
5        Extend partial solution probabilistically
6      until solution is complete
7
8    foreach ant in SelectAntsForLocalSearch() do
9      ApplyLocalSearch(ant)
10
11   EvaporatePheromones
12   DepositPheromones
13 until termination criteria met
14 end
```

Pheromones are numeric values associated to the solution components that are meant to bias the solution construction in order to improve the quality of the generated solutions. Several ants generate the solutions by an iterative approach (see listing 2.1). After this, an optional local search is applied. Next, pheromone evaporation and deposit is done. Evaporation helps to reduce the convergence-prone behavior of the algorithm. Deposit is the part where the solutions affect the pheromone values in order to bias the future solutions.

2.2.1 Choice of pheromone trails and heuristic information

Generally, there are two mechanisms of biasing the solution construction - pheromones and heuristic values.
Hereby we introduce the following components:
C - set of solution components (a combination of which can constitute a solution).
$\tau_c \in T$ - pheromones trail values for solution component bias.
$\tau'_c \in T'$ - pheromones trail values for particular purposes (as choosing the next facility to consider in QAP).
π - candidate solution.
$\eta_c \in H$ - heuristic information (constant in time).

[ACOTSPQAP brief description]

2.3 Iterated F-Race 5

[description similar to the slides of I-RACE description]

Chapter 3

Vehicle Routing Problem 5

[correct formulas below under the VRP definition in the future]

For the VRP, the *nearest neighbor* heuristic can be used. The nearest neighbor heuristic starts from a random node π_i with initial random $\pi =< \pi_1 >$. At each step it selects the node with the minimal distance d_{ij} to the current node and adds the corresponding next node $\pi_2 = j$ components to the solution. The index of the next node at each step can be computed as following.

$$nextNode = \underset{j \in N, j \notin \pi}{\operatorname{argmin}} (\pi_{ij}) \tag{3.1}$$

Chapter 4

Implementation 20-25

[framework class-level description, no class diagram due to large size (however no descriptions for every class method. it's not a documentation)]

[mention the back-bone classes that provide generality and maintaining of VRP instance generality]

[framework parameters space]

[mention unit testing???]

Chapter 5

Experimental 20-25

[public samples description, some publicly know plots]

[configuration of i-race]

[obtained result configuration for the framework]

[description of the configuration]

Chapter 6

Conclusion 4-5

6.1 Literature