Università degli studi di
Milano-Bicocca

Decision Models

Final Project

Q-ant & Q-gen

Authors:
Dario Bertazioli-847761-d.bertazioli@campus.unimib.it
Fabrizio D'Intinosante-838866-f.dintinosante@campus.unimib.it
Massimiliano Perletti-XXXXXX-m.perletti2@campus.unimib.it

June 17, 2019

Abstract

1 Introduction

The problem: the travelling salesman problem (TSP) is an algorithmic problem tasked with finding the shortest route between a set of points and locations that must be visited. In the problem statement, the points are the cities a salesperson might visit. The salesman‘s goal is to keep the distance travelled as low as possible. TSP has been studied for decades and several solutions have been theorized. The simplest solution is to try all possibilities, but this is also the most time consuming and expensive method. Many solutions use heuristics, which provides probability outcomes. It must be considered that the results are approximate and not always optimal.

Our approach: in this project we tried to apply two meta-heuristics named *Ant Colony Optimization* and *Genetic Algorithm*, implementing their "classical" version and a custom one integrating *Reinforcement Learning Algorithm*, namely *Q-learning*.

1.1 Theoretical context

1.1.1 Genetic Algorithms:

In this work we focus on two different approaches to the TSP. The former is the **Genetic Algorithm**, which we initially implement in its classical version. This algorithm is based on a biological metaphor: the resolution of a problem is seen as a competition among a population whose evolving individuals become better and better candidates solutions over time. A “fitness” function is used to evaluate each individual to decide whether it will contribute to the next generation. Then, in analogy with the biological metaphor (the gene transfer in sexual reproduction), a crossover operator is applied in order to generate the next generation of the population. This process, according to the evolutionary theory (Darwinism), should lead after a certain number of iterations to a much more fit ensemble of individuals representing "good" candidate solutions to the considered problem.

The pseudo code of the standard genetic algorithm is summarized in the Fig. 2, where Tc is the crossover rate or parameter that determines the rate at which the crossover operator is applied, Tm is the equivalent for

Algorithm 1 Genetic Algorithm

```
1: procedure Genetic(Tc,Tm,Tp,MaxIt)
2:   Pop ← GeneratePopulation(Tp)
3:   Pop ← Evaluation(Pop)
4:   for i = 1...MaxIt do
5:     Pop ← Selection(Pop)
6:     With probability Tc do:
7:     Pop ← Crossover(Pop)
8:     Pop ← Selection(Pop)
9:     With probability Tm do:
10:     Pop ← Mutation(Pop)
11:  end for
12:   return the best solution in Pop
13: end procedure
```

Table 1: Genetic Algorithm pseudocode

the mutation rate, Tp is the population size (number of chromosomes) and MaxG the number of generations used in the experiment.

With the aim to explore a new variation of the standard algorithm, we try to integrate a **Q-Learning** Algorithm in the genetic procedure in order to provide a better guideline for the initialization of the population and the crossover operation.

1.1.2 Ant Colony Optimization:

The latter kind of algorithm we implement is the **Ant Colony Optimization** (ACO) algorithm.

The procedure draws inspiration from a "real" Ant Colony. In nature, such a system is known to accomplish some difficult tasks, being beyond the capabilities of a single ant, exploiting the individuals collaborating with each other.

In particular, ACO algorithm is based on foraging behaviour of some ant species. This behaviour can be summed up as their ability to find the shortest paths between a source of food and their nest. The cooperation among the ants has inspired researchers to apply a similar collaboration based algorithm to those problems whose solutions can be formulated as a least cost path between an origin and a destination. Since most optimisation problems might have such a formulation, those kind of algorithms are pretty

interesting.

The first ACO algorithm, Ant System, was proposed by Dorigo (FIXME: add cit). It consists in a multi-agent approximate approach that it is said it can produce good-quality solutions in a reasonable time for combinatorial optimisation problems (FIXME: ADD cit dorigo (n.5 on msc thesis)). The author demonstrate the performance of this algorithm on Travelling Salesman Problem (TSP).

Regarding the basic mechanism of ACO, here follows a quick biological explaination. Ant species are almost blind, thus they interact with the environment and communicate with each other expliting the hormones they release. In partiuclar some ant species use a special kind of hormone called **pheromone**: they lay pheromone trails on the paths they explore, these traces act as stimuli and other ants belonging to the colony are attracted to follow the paths that have relatively more tracked. Due to this mechanism, an individual who is following a path because of the pheromone trail also reinforces it by dropping its own pheromone too.

Thus, the more ants follow a specific path, the more likely that path becomes to be followed by the ants in the colony (FIXME: check/add cit [8], [5], [9] msc thesis).

ACO algorithm makes use of ant-like agents called artificial ants, that construct their solutions collaboratively by sharing their experience on the quality of solutions that were generated so far. The pheromone trails play a leading role in the utilization of collective experience. The solutions are built iteratively. Artificial ants have "memory" to store the path they followed while constructing their solutions. Exploiting such a memory, typically (even though depending on the speficic class of ant colony algorithm) artificial ants do not deposit the pheromone until they have constructed their solution. Then, They determine the amount of pheromone according to the quality of their solution and upload the pheromone matrix (the data structure in which the pheromone amount for each part of the total path is stored). Automatically, the paths belonging to better solutions, receive more pheromone.

In iteratively building a solution (a total path) for a single ant, a local stochastic transition policy is typically applied, stating how to decide the next node to visit in a graph. Artificial ants make their decisions and transitions to their next state in discrete time steps, deciding whether to follow the main trails, or to random explore a new path (in our implementation, such a decision is made by a random number generation and imposing a threshold). Exploration is also encouraged by a mechanism of pheromone evaporation, which prevents the colony from getting stuck into a solution corresponding

to a (only) local optimum (note however that in real ant colonies pheromone evaporation is too slow to be a significant part of their search mechanism).

Summing up, Ant Colony Optimisation is a metaheuristic proposed to solve hard optimisation problems. The ACO metaheuristic, from a high-level view, is composed of 3 main stages:

- ConstructAntSolutions: the artificial ants construct their solutions. The transition policy controls the ants' next step to one of the adjacent nodes. Once the ants have completed their path, the quality of the current solution is evaluated, and used in the next step. The decision policy is based on following a probability distribution of the type (FIXME: add cit dorigo (5 msc thesis)):

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{I \in N_i^k} [\tau_{ij}]^\alpha [\eta_{ij}]^\beta} \tag{1}$$

  where

  - eta_{ij} indicates an heuristic value specified according to the problem (in the TSP case, is equal to $1/d_{ij}$,
  - τ_{ij} is the pheromone quantity on the path between the i-th and j-th nodes,
  - α and β are the parameters used to set the relative importance of the pheromone trail and the heuristic value. As $\alpha \to 0$, the pheromone track become less important and the ants tend to choose the closest cities, resulting in a much more "greedy" search. Viceversa, when $\beta \to 0$, heuristic values are almost ignored and only the tracks are considered in the decision making.

- UpdatePheromones: the pheromone trails are adjusted based on the latest iteration of the colony search process. Two different updates happen:

  - the pheromone evaporates according to the equation:

$$\tau_{ij} = (1 - \rho)\tau_{ij} \tag{2}$$

    where ρ is the evaporation coefficient.

  - new pheromone is deposited on the followed path. The amount of pheromone to deposit is typically decided according to the quality of the particular solutions that each path belongs to:

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{3}$$

where $\Delta\tau_{ij}^{k}$ is the pheromone increase amount deposited by the k-th ant, which can be (e.g. in Dorigo initial work) taken either as a constant, or $\Delta\tau_{ij} = 1/L_k$, where L_k is the $k - th$ ant path lenght.

However the entity of pheromone update and it's weight on how the search will be biased towards the best solution found so far is an implementation decision.

- the following equations can be combinated in:

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \Delta\tau_{ij} \tag{4}$$

(FIXME: fix pseudocode)

1.1.3 Ant-Q Metaheuristic

In order to better understanding the working mechanism of Ant-Q Metaheuristic and to give deeper insight in our implentation (still following (FIXME: add cit gamba)), let us introduce some theoretical hints for the context.

Hints on reinforcement learning: Reinforcement Learning (RL) is an (almost) unsupervised learning approach.

It consists of an **agent** who tries to learn how to reach a goal by a continuous interaction with the environment. There is an evaluation phase where the quality of agent's actions is considered and feedbacks to the agent are given in the form a numerical reward. This type of feedback is known as evaluative feedback: in contrary of supervised learning, here the agent is not explicitly told what action is the best to take in a certain situation, whereas it should try a set of possible actions and learn the best strategy yielding the most reward itself.

In some cases, the goal state (that is, the agent reaching its objective) can be obtained only after a sequence of actions: as a result the reward is delayed (FIXME: cfr section ant q delayer reward).

Summing up, and according to (FIXME: add cit 18 (msc thesis), the RL problem can be defined as the problem of an agent interacting with a complex environment trying to maximise its long-run reward over a sequence of discrete time steps.

Algorithm 2 Ant Colony Optimization

Main Algorithm

```
for generation in generations:
    create n_ants artificial ants
    for one_ant in ants:
        make_path
        compute_path_length
        update best_dist and best_path
    update pheromon matrix (local for child process)
return best_dist, best_sol
```

Make path

```
start from a vertex
add start vertex to visited nodes
for each remaining vertex:
    list the neighbors
    list the not yet visited neighb
    calculate the probability of choosing a vertex
    choice the vertex according to probability
    add the vertex to the passed list
    return the chosen vertex id
```

Update pheromon matrix

```
evaporate pheromon
for ant in ant_colony :
    for each vertex of one_ant_path :
        pheromon_matrix.increase(c_v, n_v,+1)
```

Update pheromon matrix

```
gather from MPI env all the pm matrix
if process is the parent process (rank==0):
  for each element average over the n_cores matrices.
broadcast obtained pm matrix to the child process
```

Table 2: Ant Colony pseudocode

The agent follows a **policy** to decide on its action according to the current state and conditions.

This policy is typically a stochastic function ($\pi(s, a)$) that indicates a probability of choosing an action a given a state s. Notice that agent has the possibility to change its initial policy according to new experiences in order to achieve optimal cumulative reward over time.

The value of a state $V_\pi(s)$ is defined as the expected cumulative reward that will be obtained starting from a state s and acting according to the current policy π. In the same way, the value of a pair state-action($Q_\pi(s, a)$) is the expected return obtained starting from s with action a and then following the policy. In formulas, V is defined as:

$$V^\pi(s) = E_\pi\{\sum_i \gamma^i r_{t+1,i+1 \ | s_t=s}\} \tag{5}$$

and accordingly:

$$Q^\pi(s, a) = E_\pi\{\sum_i \gamma^i r_{t+1,i+1 \ | \ s_t=s, \ a_t=a}\} \tag{6}$$

The RL problem consists in the agent trying to find the optimal policy π^* that maximizes the value functions, obtaining thus:

$$V^*(s) = max_{a \in A(s)}(Q^{\pi^*}(s, a) \tag{7}$$

however, the policy estimation can be in general a complex problem, and an optimal policy can be obtained with various algorithms, such as Policy Iteration and Value Iteration. There are also kind of learning mechanism defined as "off-policy", because they do not exploit a proper policy procedure.

Q-Learning : is an off-policy method, meaning that it updates the values iteratively basing this process on the action that gives the maximum value (that is, such an algorithm tries to directly learn Q^* instead of learning Q_π first). In figure 1 it is shown the pseudocode of the algorithm: the agent uses a so called ϵ-greedy policy, but updating the current value estimate considering the action that provides the maximum value at the successor state instead of considering the (current-)policy-suggested action.

Ant-Q Algorithm : one of the core points of this project consists in the implementation of the Ant-Q algorithm, introduced by Gambarella (FIX: add cit) in collaboration with Dorigo, attempting to ameliorate the "classic" ACO performances (in FIXME: add cit 11 master thesis).

Randomly initialise Q(s, a)
Repeat for each episode
  Initialise the current state s
  While s is not terminal state
    Choose action a at the current state s according to the policy (e.g. $\varepsilon - greedy$)
    Take action a, observe reward r and next state s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$;

Figure 1: The pseudo code of a typical Q-learning algorithm implementation.

In this approach, the pheromone update rule is borrowed from the Q-learning prassi:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma[max_{a'}Q(s^{'},a^{'}) - Q(s,a)] \ . \qquad (8)$$

In particular, equation 4 is changed into the following:

$$\tau_{ij} = (1-\alpha)\tau_{ij} + \alpha(\Delta\tau_{ij} + \gamma max_{l \in N_j^k}\tau_{jl}) \ . \qquad (9)$$

(FIXME: fix eq punctuation everywhere)

Comparing equation 9 to to the Q-learning update rule 8 , it is worth to notice that :

- equation 9 updates the pheromone value of the transition (i, j) according to the pheromone value of the next transition (j, l),
- equation 9 uses the second part of equation 8 (known as TD Error) to weight the pheromone quantity associated to the current edge with a learning rate α and a discount rate γ,
- the equation $\tau_{ij} = (1-\alpha)\tau_{ij} + \alpha(\gamma max_{l \in N_j^k}\tau_{jl})$ is used for the pheromon matrix update (namely a local update) during each path construction (of each ant), and it does not include the delayed reward $\Delta\tau_{ij}$,
- $\Delta\tau_{ij}$ is calculated according to the solution quality, as anticipated circa equation 3, and assigned in a "delayed" mode: thus the value of $\Delta\tau_{ij}$ for all i and j will be 0 while the ants apply the update rule (fixme: add rule) during their construction of the current solution. To compensate for this, the update rule 9 is reapplied at the completion of the current solution, but with the value of the next transition considered to be equal to zero. (thus uploading only with $\Delta\tau_{ij}$.

- still according to (FIXME: ADD CIT TO DORIGO ANTQ and [18]) $\Delta\tau_{ij}$ can be updated with an iteration best rule (update every single colony iteration) or global best (update based on the global best value.

2 Datasets

The datasets used in this work are taken from `https://wwwproxy.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/`, a large source of TSP datasets largely cited in literature. There are datasets of variable dimension and for everyone is also available the optimal solution so that is possible for us to compare our results with the optimal one. Every solution is available at `https://wwwproxy.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/STSP.html`. Every dataset is composed by a list of *"cities"* with two coordinates points; the only preprocessing we applied was to compute a matrix containing the distance between every point and the other ones.

3 The Methodological Approach

4 Results and Evaluation

The Results section is dedicated to presenting the actual results (i.e. measured and calculated quantities), not to discussing their meaning or interpretation. The results should be summarized using appropriate Tables and Figures (graphs or schematics). Every Figure and Table should have a legend that describes concisely what is contained or shown. Figure legends go below the figure, table legends above the table. Throughout the report, but especially in this section, pay attention to reporting numbers with an appropriate number of significant figures.

5 Discussion

The discussion section aims at interpreting the results in light of the project's objectives. The most important goal of this section is to interpret the results so that the reader is informed of the insight or answers that the results provide. This section should also present an evaluation of the particular approach taken by the group. For example: Based on the results, how could the experimental procedure be improved? What additional, future work may be warranted? What recommendations can be drawn?

6 Conclusions

Conclusions should summarize the central points made in the Discussion section, reinforcing for the reader the value and implications of the work. If the results were not definitive, specific future work that may be needed can be (briefly) described. The conclusions should never contain "surprises". Therefore, any conclusions should be based on observations and data already discussed. It is considered extremely bad form to introduce new data in the conclusions.

References

The references section should contain complete citations following standard form. The references should be numbered and listed in the order they were cited in the body of the report. In the text of the report, a particular reference can be cited by using a numerical number in brackets as