Swarm intelligence: Ant colony optimization project

Samuel Buchet: 000447808

June 2017

1 Introduction

In this project, two anto colony optimization algorithms are implmented to solve the Closest String Problem. The two algorithms implemented are the Max Min Ant System algorithm (MMAS) and the Ant Colony System algorithm (ACS). In the following report, the two algorithms are first described. After that, the two algorithms are compared. Finally, a local search is used to improve the results.

2 Implementation of the algorithms

In this section, the Max Min Ant System and the Ant Colony System algorithms are described. These two agorithms also use a heuristic information which is explained below.

2.1 Heuristic information

In [1], no heuristic information is used. However, most of the ant colony algorithms use this kind of information. In addition, it can be easily computed for a lot of problems. It has been decided to define one for the closest string problem.

The goal of the closest string problem is to minimize the maximum hamming distance between the solution and the set of strings. To minimize the distance between the solution and the set of strings, a greedy decision consists in adding into the solution the character which appears the most in the set of strings at a given position. Indeed, by using this character, the distance might be equal to 0 at this position. For each character at each position of the string, a greedy score can be defined as the frequency of the character.

However, the frequencies might be very close to each other. As a result, if this score is used in the probabilities, the probabilities of getting the best greedy character would be very low. To solve this problem, the exponential function is applied to the score after scaling it. The final score of character j at position i is equal to: $score_{ij} = \frac{exp(5*frequency_{ij})*1.5}{max_j(frequency_{ij})}$ where $frequency_{ij}$ is the number of occurences of the character j at position i in the set of string of the problems. The parameters have been chosen after testing different values.

2.2 Basic ant system algorithm

Both variants of the ant colony algorithms implemented in this project rely on the same basis. At each iteration, the general algorithm builds n solutions (with n equal to the size of the population) and then updates the pheromones with the formula: $\tau_{ij}(t) = (1-\rho) * \tau(t-1) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$ where $\tau_{ij}(t)$ is the amount of pheromone at position i for character j after t iterations, ρ is the evaporation rate and τ_{ij}^{k} is the quantity of pheromone deposited by ant k if this ant have chosen character j at position i.

As seen in [1], the amount of pheromone deposited by the ants is equal to $1 - \frac{HD}{m}$ where HD is the maximum hamming distance of the ant and m is the length of the strings of the problem. This quantity has been used in the two implementations of this project.

To build a solution, each character is chosen according to a probabilty. The probability of choosing the character j at position i is equal to $\frac{greedyScore_{ij}^{\alpha} * \tau_{ij}^{\beta}}{\sum probas}$ where $greedyScore$ is the greedy score described previously and τ_{ij} is the amount of pheromone at position i for the character j.

2.3 Max Min Ant System

In the Min Max Ant System algorithm, the pheromones are constrained with an upper and a lower bound. At the begining of the algorithm, the pheromones are inizialised to $+\infty$. After each pheromone updates, the pheromones are checked. If it exeeds the lower or the upper bound, the pheromone is re-assigned to this bound.

In this implementation, the upper bound is equal to $1 - \rho * \frac{D_{best}}{m}$ where D_{best} is the value of the best solution found so far and m is the length of the string. The lower bound of the pheromones is equal to $\frac{up}{a}$ where up is the upper bound and a is a parameter. These two bound are updated each time a better solution is found.

In this Max Min Ant System implementation, the best solution of a population generation is the only one which deposits pheromones. An addition component is used to prevent the algorithm from converging. If a convergence is detected, the pheromones are re-initialised to the upper bound. The convergence is detected by a certain number of iterations without any improvements of the best solution found. This number of iterations is equal to $iterations * convRate$ where $iterations$ is the maximum number od iterations allowed to the algorithm and $convRate$ is a parameter, between 0 and 1 which determines the percentage of the total time allowed to wait before pheromones re-initialisation.

2.4 Ant Colony System

The Ant Colony System algorithm is usefull to control the trade off between exploration and intensification. In this variant, a additional rule is can be used to build a solution. With probability q_0, the artificial ant chooses the character j at position i which maximizes $\tau_{ij} * \eta_{ij}^{\beta}$, where τ_{ij} is the amount of pheromones at position i for the character j, η_{ij} is the heuristic information for the same character and q_0 is a new parameter of the algorithm. With probability $1 - q_0$, the previous rule is applied (biased exploration).

The pheromones update, which takes place after building the solutions of the population, is also modified. Only the best ant so far deposits pheromones. The amount of pheromones deposited is the same as before, but it is multiplied by the parameter ρ. To finish, a new local update rule of the pheromones is used to introduce diversification. During the construction of the solutions, for each decision chosen by the ants, the pheromones corresponding to that decision is updates as following: $\tau_{ij} = (1 - \rho) * \tau_{ij} + \rho * \tau_0$, where τ_{ij} is the amount of pheromones for this decision and τ_0 is the initial amount of pheromones.

3 pheromones and probas

To obtain a completely greedy solution, the alpha should be very big and negative. With this settings, the pheromones should be greater or equal to one, otherwise the result is too big.

4 Parameter tuning

MaxMin gives good results, but still to improve. main problem: resetting pheromone has no effect maybe because the max limit is too low. Something weird with the pheromones in maxmin : if it does not deposit pheromones, it still improve well.

ACS: Initially, high initial pheromones bad result Then try with the value 1, much better results, pheromones really used, very close to the best solution. Always improving

5 Local search

if the value is not close to the optimal, the local search improves a lot. Otherwise, it improves just a little.

References

[1] Simone Faro and Elisa Pappalardo. Ant-CSP: An Ant Colony Optimization Algorithm for the Closest String Problem, pages 370-381. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.