Swarm intelligence: Ant colony optimization project

Samuel Buchet: 000447808

June 2017

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.

1 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.

1.1 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.

1.2 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.

1.3 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.

1.4 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 updated 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.

2 Parameter settings

Parameter settings is a crucial step. The ant colony optimization algorithms contain a lot of parameters and they are efficient only for a narrow range of values.

2.1 Process

To make the tuning easier, the assumption that the parameters are not completely dependent is made. This means that if a value which gives very good solutions is found for a parameter, the solution qualities will not change a lot by slightly modifying the other parameters. With this assumption, a good setting can be found by modifying one parameter at a time. If a significant improvement is found with a certain parameter, the other parameters can be re-tuned to continue improving the algorithm. This process can be repeated with multiple instances until no significant changes are found anymore.

To tune the parameters, it is also possible to display the informations of the algorithm during the execution of some instances. For example, during the tuning phase of the algorihtms, the best solution found so far was displayed each time a better solution was found. By doing this, it is possible to detect if the algorithm converges and if the total budget is used. If no improvement occurs long before the end of the execution, this could mean that the algorithm has converged. It can be checked by displaying the pheromones and it can be resolved by changing the ρ value.

To analyse the performance of the tuning, the solution returned by the algorithm can also be compared to a greedy solution. The greedy solution can be generated by taking the character which maximizes the heuristic score described in section 1. If the result of the ant colony algorithm is not better than the result of the greedy one, it could mean that the ant colony algorithm completely rely on the heuristic information and that the α and β parameters should be modified.

2.2 Max Min settings

Parameter	Value
$ants$	10
α	1.8
β	2
ρ	0.12
a	60
$convRate$	0.16

The number of ants has been set to 10 in this algorithm, as done in [1]. With the same number of iterations, a higher value could give better results but it would require more evaluations. The ρ parameter should seem to be high comparing to [1] (0.03). However, the pheromones bounds and the pheromone re-initialisation prevent the algorithm form converging. In order to properly use the pheromones, the a parameter should be quite high. However, higher values did not give better results, which can be caused by the very low value of the upper bound. To finish, the $convRate$ parameter (percentage of the maximum number of iterations to wait before) has been adapted to keep enough time for intensification.

2.3 Ant Colony System settings

Parameter	Value
$ants$	10
α	1.2
β	2.5
ρ	0.015
q_0	0.6
$initPheromones$	0.026

In this algorithm, the number of ants is the same as before. We can see that the value of ρ is very low compared to the previous settings. This can be explained by the fact that there is no

bounds on the pheromones and no re-initialisation. The initial amount of pheromone is also very low compared to the previous one ($+\infty$ for the max min ant system). This value can be related to the percentage of the initial pheromone value deposited by all the ants during the construction of the solutions. Finally, the q_0 parameter is greater than 0.5 which means that the intensification rule is chosen more often.

3 Comparison of the algorithms

To compare the algorithms, the relative percentage deviation has been computed with the mean of the 10 executions of each algorithm on each instance.

3.1 Relative Precentage Deviations

Instance	MMAS rpd	ACS rpd
20-10-10000-1-9	0.005	0.051
20-20-10000-1-7	0.134	0.220
20-30-10000-1-5	0.150	0.211
20-40-10000-1-3	0.209	0.266
20-50-10000-1-1	0.308	0.357
2-30-10000-1-9	0.327	0.605
2-30-10000-2-8	0.356	0.611
2-40-10000-1-7	0.766	1.034
2-40-10000-2-6	0.546	0.724
2-50-10000-1-5	0.583	0.810
2-50-10000-2-4	0.696	0.914
4-20-10000-1-2	0.208	0.396
4-20-10000-2-3	0.073	0.109
4-30-10000-1-1	0.205	0.334
4-30-10000-2-0	0.198	0.262
4-40-10000-1-9	0.395	0.498
4-40-10000-2-8	0.130	0.164
4-50-10000-1-7	0.419	0.542
4-50-10000-2-6	0.040	0.056

The table bellow shows the min, the max, the mean and the standard deviation of the relative percentage deviations.

algorithm	min	max	mean	sd
MMAS	0.005	0.766	0.303	0.219
ACS	0.051	1.034	0.430	0.292

As we can see, the value for the Min Max algorithm is always lower than the value of the Ant Colony System algorithm.

To conclude that there is a statistically significant difference between the two algorithms, we can perform a Wilcoxon rank-sum test. The test performed with a significance level equal to 0.05 returns a p-value equal to 0.154. The p-value is greater than the significance level, which means that the null hypothesis is not rejected. Thus, we cannot conclude that there is a statistically significant difference between the rpd of the two algorithms. There is no algorithm which is statistically better than the other one. However, regarding the results on the table bellow, we could recommend the Max Min algorithm, without any guarantee.

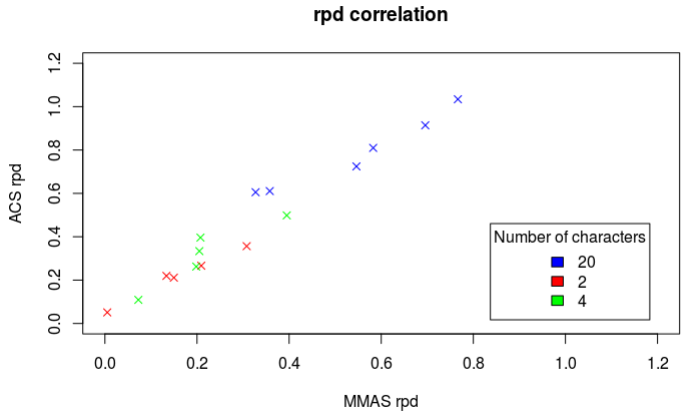

Figure 1: Correlation plot between the arpd's

3.2 Correlation of the algorithms

The plot 1 shows the correlation between the rpd's of the two algorithms for all instances. Each color corresponds to the instances with a same number of characters in the alphabet.

We can see that the rpd's of the rwo algorithms are correlated. This can be confirmed with a spearman test. With a significance level equal to 0.05, the test returns a p-value equal to $8.085 * 10^{-6}$ which means that the results are correlated. We can see on the graph that the instances with 20 characters are the most difficult. The correlation shows that the more an instance is difficult for one algorithm, the more this instance will be difficult for the second algorithm. This observation sows that the algorithm have a similar behaviour on the problem.

3.3 Convergences of the algorithms

The two plots 2 and 3 can be used to compare the convergence of the two algorithms. These plots have been done using the same tuning with a number of iterations equal to 2000 to have enough time to observe the convergence.

On both plots, we can see that the the Max Min algorithm is able to reach a better solution. However, on the first plot, we can see that at the begining of the run, the Ant Colony System algorithm find better solutions than Max Min Ant System. This result is not clear on the second plot but it could mean that for some instances, if the budget is very limited, it would be preferable to use the Ant Colony System Algorithm.

On the first plot, we can see that the convergence of Ant Colony System is faster, which can be explained by the fact that it does not have a pheromone re-initialisation process like Max Min Ant System. However, on the second plot, we can see that the convergence of the Ant Colony System algorithm is slower.

4 Local search

To improve the result of the algorithm, it is possible to execute a local search. In this project, the local search is applied to the Max Min Ant System Algorithm.

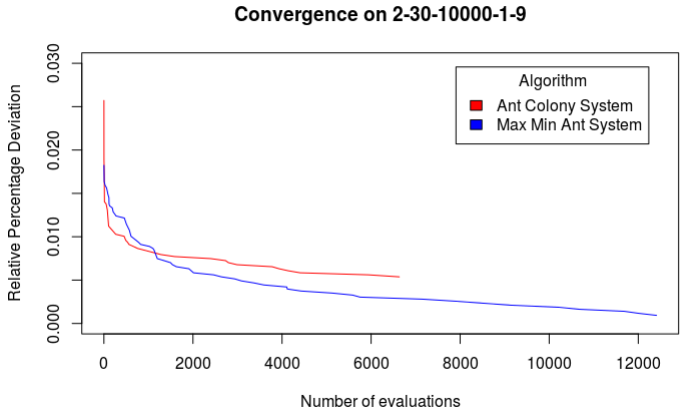

Figure 2: Convergence on 2-30-10000-1-9

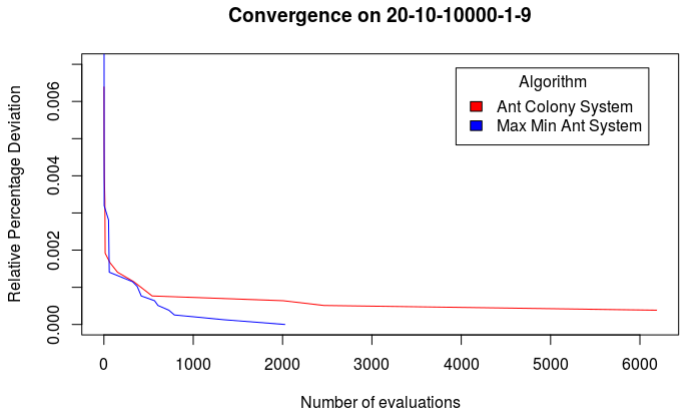

Figure 3: Convergence on 20-10-10000-1-9

4.1 Principle

The principle of a local search is to perform few modifications on a solution. These modifications create new solutions which are calle neighbours. These solution can be evaluated and hopefully replace the best solution found so far. Once a better solution is found, the process can be executed again on this new solution.

4.2 Application to the Closest String Problem

The neighbourhood chosen is the set of solutions that differs with only one character from the initial solution. The local search algorihtm implemented iterates over all the neighbours and select the best one. (best improvement rule) If the best neighbour is better than the best solution found so far, the process is restarted with this new solution.

To implement the local search efficiently, the evaluation of the solution needs to be fast. In this problem, this is possible by keeping the distances of each strings in the memory and comparing the new character of the solution to the previous one to update the distance ifnecessary. However, the execution of the local search after each improvement remains slow. For this reason, the local search is only applied at the end of the algorithm, on the best solution found.

4.3 Comparions with the local search

Conclusion

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.