Evolving Strategies for Updating Pheromone Trails: a Case Study with the TSP

Jorge Tavares[1] and Francisco B. Pereira [1,2]

CISUC, Department of Informatics Engineering, University of Coimbra[1]
Polo II - Pinhal de Marrocos, 3030 Coimbra, Portugal
ISEC, Quinta da Nora, 3030 Coimbra, Portugal[2]
jorge.tavares@ieee.org, xico@dei.uc.pt

Abstract. Ant Colony Optimization is a bio-inspired technique that can be applied to solve hard optimization problems. A key issue is how to design the communication mechanism between ants that allows them to effectively solve a problem. We propose a novel approach to this issue by evolving the current pheromone trail update methods. Results obtained with the TSP show that the evolved strategies perform well and exhibit a good generalization capability when applied to larger instances.

1 Introduction

Ant Colony Optimization (ACO) draws its inspiration from pheromone-based strategies of ant foraging. Initially, it was conceived to find the shortest path in the well-known Traveling Salesman problem (TSP), but soon it was applied to several different types of combinatorial optimization problems [1]. Examples of situations addressed include both static and dynamic variants of academic and real world problems. Usually, the problem is mapped into a fully connected graph. When seeking for a solution, ants deposit pheromone while traveling across the graph edges, thus creating a virtual trail. A solution to the problem will emerge from the interaction and cooperation made by the ants. Different types of ACO architectures were proposed in the literature [1], with differences, e.g., in what concerns the way ants deposit pheromone in the graph edges. Designing new variants of ACO algorithms is an active area of research, as this may allow the application of these methods to new situations and/or enhance its effectiveness on problems that it usually addresses. Typically, the design and extension of the existing algorithms is carried out manually.

In this paper our approach will be different. We aim to automatically develop and improve the current components by using Evolutionary Algorithms (EA). Our framework relies on a Genetic Programming algorithm (GP) [2] to evolve the way an Ant algorithm updates its pheromone trails. We investigate the evolution of different strategies and study how they perform when compared to standard methods. Different instances of the TSP will be used to access the effectiveness of the proposed approach.

The paper is structured as follows. In section 2 we present the system to evolve the pheromone trails update strategies. Section 3 contains the experimentation and analysis, followed by a discussion. Finally, in section 4 we summarize the main conclusions and highlight directions for future work.

2   Evolving Pheromone Trail Update Methods

There are many research efforts for granting bio-inspired approaches the ability to self adapt their strategies. On-the-fly adaptation may occur just on the parameter settings or be extended to the algorithmic components. One remarkable example of the first type of self-adaptation is the well-known 1/5 success rule used to control the mutation strength for the (1+1)-ES. In what concerns the adaptation of the optimization algorithm, Diosan and Oltean recently proposed an evolutionary framework that aims to evolve a full-featured EA [3].

As for the Swarm Intelligence area, there are also some reports describing the self-adaptation of parameter settings (see, e.g., [4, 5]). Still, there are a couple of approaches that resemble the framework proposed in this paper. Poli et. al [6] use a GP algorithm to evolve the update equation that controls particle movement in a Particle Swarm Optimization (PSO) algorithm. Diosan and Oltean also did some work regarding PSO structures [7]. Also, Runka [8] applies GP to evolve the probabilistic rule used by an ACO algorithm to select the solution components in the construction phase.

2.1   Overview of the System

The framework to evolve pheromone trails update strategies is composed of two main components: a Genetic Programming (GP) engine and an Ant System (AS) algorithm. The main task of the GP component is to evolve individuals that encode effective trail update strategies, whereas the AS is required to assign fitness to each generated solution. The GP engine adopts a standard architecture: individuals are encoded as trees and ramped half-and-half initialization is used for creating the initial population. The algorithm follows a steady-state model, tournament selection chooses parents and standard genetic operators for manipulating trees are used to generate descendants. As for the AS framework, it closely follows the variants proposed in [1]. It allows the application of the standard AS, Elite AS (EAS) and Rank-based AS for the TSP, as described in the aforementioned book. The activation of the pheromone update methods was defined in a way to allow an easy integration with the GP engine: when an individual (i.e., an evolved pheromone trail update strategy) needs to be evaluated, the GP executes the AS algorithm for a given TSP instance. The result of the optimization is assigned as the fitness value of the GP individual.

A key decision in the development of the framework is the definition of the function and terminal sets used by the GP, as they will determine which components can be used in the design of pheromone update strategies. In this paper our aim is to show that the proposed approach is able to evolve such strategies.

Therefore, to validate our ideas we keep the definition of the function and terminal sets as simple as possible. Our choice is to provide these sets with ingredients that allow the replication of standard AS and EAS methods. By providing the self-adaptive algorithm with these components we ensure that a valid strategy exists and, at the same time, we explicitly verify if the evolutionary process is able to determine it. The function and terminal sets are composed by:

- *(prog2 p1 p2)* and *(prog3 p1 p2 p3)*: They allow the sequential execution of two or three functions/terminals. The result of the last one is returned;
- *(evaporate rate)*: Runs the standard evaporation formula with a given *rate*.
- *(deposit ants amount)*: *ants* deposit a given *amount* of pheromone. The parameter *ants* can be an array of ants or a single one.
- *(all-ants)*, *(best-ant)*, *(rho)*: They return: 1) the array with all the ants; 2) the best ant found so far in a run; 3) a fixed learning rate.
- *(integer)* and *(real)*: Ephemeral constants.

The evaporation and deposit formulas are the ones defined in the literature [1], with the exception of the given parameters. Since standard GP needs the closure property [2], all the functions and terminals are protected to ensure that no invalid combination can be formed.

3 Experiments and Analysis

Selected instances from the TSPLIB[1] are used in the experiments. For all tests, the GP settings are: Population size: 100; Maximum tree depth: 5; Crossover rate: 0.9; Mutation rate: 0.05; Tourney size: 3. For the AS algorithms we used the standard parameters found in the literature [1]. The number of runs for all experiments is 30. To determine the existence of statistical differences in the results, we apply the Wilcoxon rank sum and the Kruskalwallis tests ($\alpha = 0.05$).

3.1 Evolution of the Update Strategies

In the first set of experiments we aim to detect the evolution of feasible update strategies. A crucial issue that needs to be addressed is the evaluation step of each evolved strategy, as it requires the execution of an AS algorithm. Two parameters, the number of runs and the number of iterations per run, define the optimization effort of the AS and can have a major impact on the behavior of the self-adaptive framework. On the one hand, if we grant the AS a small optimization period to evaluate a GP individual, then it might not be enough to correctly estimate the quality of an update strategy. On the other hand, a longer evaluation period can increase computational costs to insupportable levels. For the eil51 instance (a TSP instance with 51 cities adopted for these experiments), AS variants find a near-optimal solution within approximately 100 iterations. With this value in mind, we defined three different configurations for
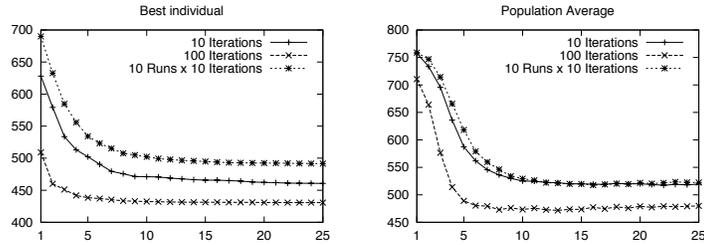
---

[1] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Fig. 1. Evolution plots with generations on the x axis and fitness values on the y axis.

the evaluation component of the GP individuals: 1) a single AS run with 100 iterations; 2) a single AS run with 10 iterations; 3) 10 AS runs with 10 iterations. Configuration 1 grants enough time for a strategy to find good solutions. The second one will test the increase of evolutionary pressure, even knowing that finding good solutions in 10 iterations is hard. In the third case we want to see the effect of multiple runs.

The left plot on Figure 1 shows the evolution of the mean best fitness (MBF - average of the best individuals generated by the GP over 30 runs) for the three evaluation configurations. The plot on the right displays the evolution of the average fitness of the population. It is clear that evolution proceeds as in a typical EA. Although individuals from the early stages of the optimization are unable to find good solutions for the TSP, after some time the GP starts to discover effective update strategies (the best solution for eil51 has length 426).

By comparing the three evaluation configurations, we conclude that single runs perform better, particularly when using 100 iterations. A single run with 10 iterations also enables the discovery of sporadic good quality solutions, but, in this case there is a larger deviation from the global optimum. This is consistent with the expected outcome. The time for the AS to build a good solution is short and the additional evolutionary pressure is not enough to eliminate the bad solutions. Using multiple runs attained the worst results. The reason is simple: the fitness assigned to the update strategy is the average of 10 solutions and, initially, most of these are bad solutions. The number of iterations to counterbalance this effect would need to be longer. In the subsequent analysis we will not consider the best tree from the multiple runs configuration since it performs worse than the other strategies.

3.2 Validation and Analysis of the Evolved Update Strategies

To validate the best evolved solutions we will compare their optimization performance with existing AS variants (AS and EAS). We selected the best three trees evolved by the system in the experiments described in the previous section: GP10 is the best update strategy found in 10 iterations, whereas GP100A and GP100B are the best ones found in 100 iterations. The TSP instance used in

Table 1. Comparison of the best solution between the Strategies, with 100 iterations and 1000 iterations for 30 runs.

| Strategies	Iterations	Best	Worst	Mean Best Fitness	Deviation	Branching
AS	100	442	471	459.10	7.46	5.25
EAS	100	432	467	447.37	9.76	3.54
GP10	100	**426**	469	446.17	8.83	2.77
GP100A	100	**426**	**456**	**443.00**	9.50	2.50
GP100B	100	**426**	472	445.70	10.58	2.96
AS	1000	431	454	444.03	5.06	5.17
EAS	1000	428	451	440.40	5.72	3.45
GP10	1000	**426**	448	435.40	5.89	2.75
GP100A	1000	**426**	446	435.23	5.35	2.21
GP100B	1000	**426**	**443**	**434.20**	4.78	2.82
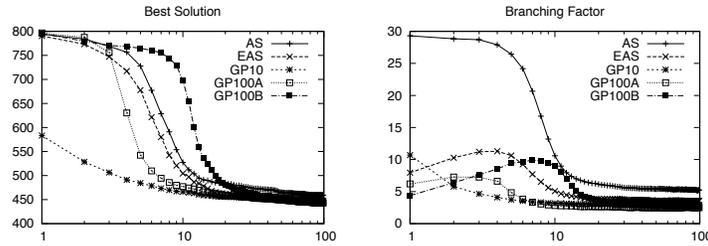
Fig. 2. Plots showing the evolution of the best solution and the branching factor for 100 iterations (x axis). The same pattern is observed with 1000 iterations.

these tests is the same one adopted for the evolution of the update strategies. Table 1 contains the results obtained by the five AS strategies (3 evolved and 2 standard) for two different optimization scenarios: the upper part of the table shows the outcomes obtained when the AS was allowed to run for 100 iterations and the lower part is from experiments that ran for 1000 iterations. In the last column we include the branching factor (with $\lambda = 0.05$), a measure to determine the convergence of the pheromone matrix. Results clearly show that all evolved solutions perform better than the standard methods. The optimal solution was always found by the evolved strategies, whereas AS and EAS were unable to discover it. An inspection of the MBF values confirms that the evolved strategies tend to achieve better results. This trend is more evident in longer runs.

The evolved strategies perform well and are competitive with the standard variants. Even with restrict function and terminal sets it is possible to evolve behaviors that are not exact copies of the ones adopted by standard AS. The evolved strategies are elitist in nature (this is not surprising given the components at the disposal of the GP algorithm), but there is some originality in the way they manipulate the pheromone matrix. Figure 2 confirms the distinctiveness of the evolved strategies. They mostly resemble the EAS, especially with regard to the search space exploration (given by the branching factor). Note
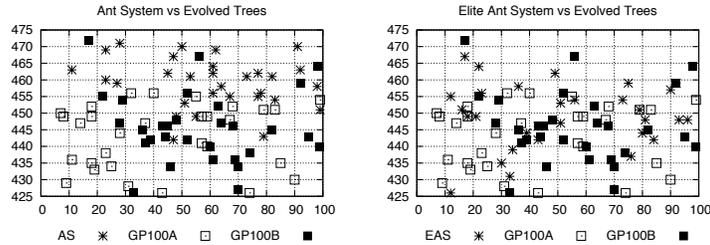
Fig. 3. Scatter plots of the best solution found in 30 runs using 100 iterations.

one important difference: tree GP100B. This solution starts to act more like AS and, after some time, quickly converges to better solutions. For example, with 1000 iterations the MBF achieved by GP100B is lower than that of all other approaches. The reason for this is related to how the strategies work. GP10 and GP100A apply a strong evaporation procedure and then allow the best ant to deposit pheromone. This effectively *cleans* the pheromone matrix with the exception of the best ant trails. GP100B carries out a lighter evaporation of the matrix. Since most of the pheromone is deposited by the best ant, it takes some time for this extra deposit to take effect. When that happens the convergence to the best solution becomes faster. For 100 iterations, we only find statistical significant differences between AS and the evolved strategies, whilst, for 1000 iterations, significant differences also exist with EAS.

To conclude, Figure 3 presents scatter plots of the best solutions found in the 30 runs (y axis) with the iteration when they were built (x axis). We compare both GP100A and GP100B strategies with the AS and EAS. The plots show that most of the best solutions found by the evolved strategies have better quality than the ones produced by the AS. There is a clear separation between both evolved update methods and AS. This is not so evident when comparing with EAS. Although there are more solutions from the evolved strategies in the lower left corner of the plot (which indicates better solutions found in less time) than solutions from EAS, overall the dispersion is similar. This reinforces the elitist nature of the evolved strategies and that in spite of being similar, they still present different performance values.

3.3 Using Different Function and Terminal Sets

Even though the self-adaptive framework evolved moderately different strategies, the function and terminal sets used in the previous sections allow the exact replication of the standard AS and EAS trail update strategies. Now we will slightly change this, as we want to test the ability of the system to evolve effective solutions with component sets that do not allow to write the standard methods. The changes are simple: 1) we remove the function *(all-ants)* thus removing the possibility of all ants to deposit pheromone; 2) we add the function *(rank-ants*
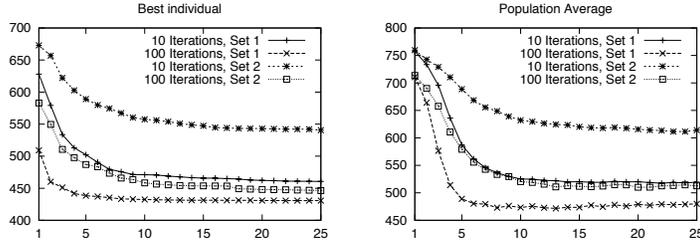
Fig. 4. Evolution with the new sets, plotting generations (x axis) and fitness (y axis).

Table 2. Comparison of the best solution between the Strategies, with 100 iterations and 1000 iterations for 30 runs.

Strategies	Iterations	Best	Worst	Mean Best Fitness	Deviation	Branching
Rank-AS	100	429	460	439.87	6.66	2.00
GP10s2	100	437	468	453.70	8.62	2.00
GP100As2	100	**426**	443	432.20	5.13	2.00
GP100Bs2	100	**426**	**437**	**431.23**	2.68	2.01
Rank-AS	1000	428	449	436.63	5.43	2.00
GP10s2	1000	435	490	454.73	13.06	2.00
GP100As2	1000	427	**440**	431.73	3.56	2.00
GP100Bs2	1000	**426**	442	**430.87**	3.52	2.00

n) which ranks the ants by solution quality and returns the best n ants. With these changes, the evolution of any of the standard AS variants (AS, EAS and Rank-AS) is no longer possible. We also want to see if the evolved strategies can achieve the same level of success as the previous ones, and if the evolved behaviors compare to the standard Rank-AS.

We fed the GP with the new sets and repeated the experiments described in section 3.2 (for configurations 1 and 2). The plots from figure 4 present a comparison between the results achieved by both sets. A brief perusal of the charts reveals that the evolution of update strategies is slower with the new test set, particularly when 10 AS iterations are used to evaluate GP individuals. On the contrary, the number of GP runs that evolved strategies with the ability to find the optimal solution increased from 2 to 6.

Once again we performed some additional tests to measure the optimization performance of three evolved strategies: GP100As2, GP100Bs2, GP10s2. The first two are examples of strategies that discovered the optimal solution when 100 iterations were performed, whereas the last is the best strategy found with 10 iterations. Table 2 presents an overview of the optimization results. The three evolved strategies and also the standard Rank-AS were applied to solve eil51, both for 100 and 1000 iterations. The outcomes confirm that evolved strategies (with the exception of GP10s2) are effective. Results are even slightly better than those achieved by the evolved strategies described in section 3.2. The comparison with the standard methods is also favorable to the evolved strategies.
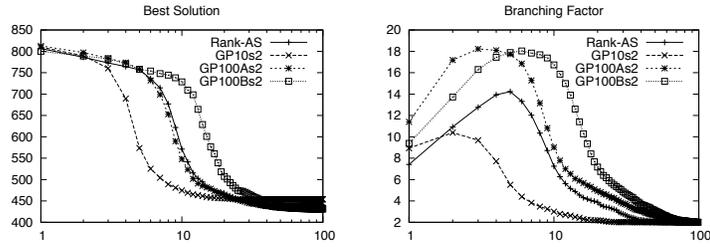
Fig. 5. Plots showing the evolution of the best solution and the branching factor of the second function and terminal sets for 100 iterations (x axis). The same pattern is observed with 1000 iterations.
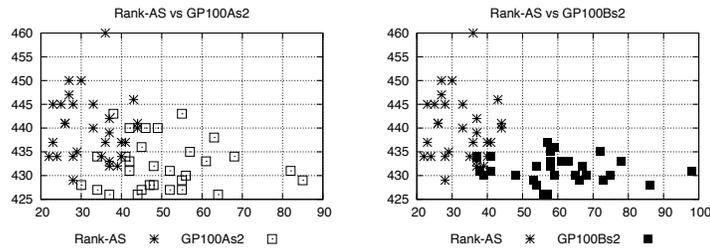

Fig. 6. Scatter plots of the best solution found in 30 runs using 100 iterations.

GP100As2 and GP100Bs2 act on a similar way. They pick the best 8 to 10 ants and allow them to deposit a large amount of pheromone. The evaporation step is performed before and/or after. No specific deposit for the best ant is done. This is a surprising consequence. The standard Rank-AS allows all the ants to deposit, picks the 6 best ants and allows them to deposit a weighted sum of pheromone and, finally, the best ant deposits the largest amount of pheromone. In the best evolved strategies, the deposit of the best ant disappears, therefore removing some greediness. For both 100 and 1000 iterations, we find statistically significant differences between the evolved trees and all the standard methods.

Figure 5 plots the behavior of these new strategies and the Rank-AS. The curves are similar in shape, especially in terms of best solution quality, but different in size. The evolution of the branching factor is interesting since it shows a larger exploration ability by the ants adopting GP100As2 and GP100Bs2, followed by a gradual convergence to the best solution. This demonstrates that the evolved strategies keep more options during the initial stage of the search and converge to a good solution in the later steps. The scatter plots in Figure 6 help us to see this. They tell us that Rank-AS converges faster but is unable to reach the best solutions. Evolved strategies take longer, but are consistently able to find better solutions.

Table 3. Results with larger TSP instances using 100 iterations for 30 runs.

Strategies	Instance	Size	Best	Worst	Mean Best Fitness	Deviation
AS	kroA100	100	22759	24258	23649.90	367.90
EAS			22339	24646	23457.07	535.40
Rank-AS			22004	24285	23113.00	544.00
GP100A			22062	25165	23310.80	726.26
GP100As2			**21632**	**23994**	**22778.80**	560.45
AS	d198	198	17487	18533	18098.37	242.21
EAS			17515	18871	18138.77	369.62
Rank-AS			16922	18310	**17486.50**	309.34
GP100A			**16698**	18689	17859.80	408.62
GP100As2			16853	**18179**	17696.13	276.97
AS	lin318	318	49075	52449	50742.73	780.21
EAS			47688	54174	50919.20	1553.24
Rank-AS			45804	**49505**	47665.87	962.95
GP100A			45323	51152	48392.23	1244.16
GP100As2			**44534**	49782	**47037.73**	1034.50
AS	pcb442	442	64535	69335	67244.40	991.15
EAS			60140	68257	64274.20	2435.61
Rank-AS			60637	70377	66843.77	2070.65
GP100A			**58858**	**67038**	**63129.54**	2215.41
GP100As2			65338	71367	68808.16	1477.87

3.4 Generalization to other TSP Instances

The previous sections dealt with the ability of the self-adaptive framework to evolve an effective strategy for updating pheromone trails. All tests were done with a single instance and this raises the question whether the evolved strategies are general enough so that they can be useful in other situations. To address this issue, we applied the evolved strategies to solve larger instances. By comparing the results of the standard approaches and the evolved methods we can obtain some evidence regarding the generalization ability of the system.

Table 3 contains the results obtained by GP100A and GP100As2 on four larger TSP instances (other evolved strategies follow a similar pattern). In short, the evolved strategies generalize. For most cases they attain the best results in terms of absolute quality and MBF. Although the differences are not large, it proves that a strategy evolved from a particular instance can be used to solve different and larger ones. There are significant statistical differences between the evolved strategies and the other methods for all instances with just a few exceptions (for kroA100 between both evolved strategies and EAS; and for pcb442 between EAS and GP100A).

3.5 Discussion

The evolved strategies are effective for solving the instance for which they were evolved, while they also exhibit a good generalization capability. However, it must be pointed out that the system uses high-level function and terminal sets, which resemble the actual standard AS methods. This choice rules out the possibility of evolving strategies that strongly deviate from standard ones. Moreover,

the evolution is not too difficult as the key components are provided and GP just needs to find the proper arrangement. Nevertheless, the results show that evolution did not converge to the standard methods (in spite of, for example, the first sets permitting it) and found different designs in structure and in parameters. This indicates an open space for improvement of the actual methods used in Ant-based algorithms. The study described in this paper is a first approach to this effort. Still, the study and use of high-level sets is important. Finally, as an example, we show one of the evolved trees GP100A:

```
(prog2 (prog3 (evaporate 0.064867854)
              (deposit (all-ants) 0.064867854)
              (prog3 (evaporate (rho))
                     (evaporate 0.6832942)
                     (deposit (best-ant) 0.699499)))
       (evaporate (rho)))
```

4 Conclusions

In this paper we proposed a system to evolve strategies for updating pheromone trails in the standard Ant System architecture. The TSP was used as a test case and two different function and terminal sets were considered. The evolved strategies have proven to perform good when compared to the standard AS, EAS and Rank-AS update methods. Moreover, they showed a generalization capability when applied to different and larger instances. In the future, we plan to design and use different function and terminal sets to evolve strategies with a non-elitist and ranking behavior. Additionally, the generalization of the evolved strategies to different problems will be addressed.

References

1. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press (2004)
2. Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk` (2008) (With contributions by J. R. Koza).
3. Diosan, L., Oltean, M.: Evolutionary design of evolutionary algorithms. Genetic Programming and Evolvable Machines **10** (2009) 263–306
4. Botee, H., Bonabeau, E.: Evolving ant colony optimization. Advances in Complex Systems **1** (1998) 149–159
5. White, T., Pagurek, B., Oppacher, F.: ASGA: Improving the ant system by integration with genetic algorithms. In: Proc. of the Third Genetic Programming Conference, Morgan Kaufmann (1998) 610–617
6. Poli, R., Langdon, W.B., Holland, O.: Extending particle swarm optimisation via genetic programming. In: EuroGP 2005 Proceedings. (2005) 291–300
7. Diosan, L., Oltean, M.: Evolving the structure of the particle swarm optimization algorithms. In: EvoCOP 2006 Proceedings. (2006) 25–36
8. Runka, A.: Evolving an edge selection formula for ant colony optimization. In: GECCO 2009 Proceedings. (2009) 1075–1082