Comparative Study: ACO and EC for TSP

Urszula Boryczka[1] and Rafał Skinderowicz[1] and Damian Świstowski[1]

[1] University of Silesia, Institute of Computer Science, Sosnowiec, Poland,
e-mail: uboryczk@us.edu.pl

Abstract. Evolutionary Computing (EC) and Ant Colony Optimization (ACO) apply stochastic searching, parallel investigation as well as autocatalitic process (or stigmergy) to solve optimization problems. This paper concentrates on the Traveling Salesman Problem (TSP) solved by evolutionary and ACO algorithms. We consider the sets of parameters and operators which influence the acting of these algorithms. Two algorithmic structures emphasizing the selection problem are discussed. We describe experiments performed for different instances of TSP problems. The comparison concludes that evolution, which is exploited especially in evolutionary algorithms, can also be observed in the performance of the ACO approach.

1 Introduction

The work focuses on the application of evolutionary (EC) and ant colony (ACO) approaches to the Traveling Salesman Problem (TSP). It was the first application of the ACO algorithm to a path optimization problem. The TSP was chosen for many reasons: it is an NP-hard optimization problem and a standard test–bed for new algorithmic ideas and good performance on the TSP is often taken as a proof of their usefulness. It is a problem to which ACO algorithms are easily applied and it is an NP-hard optimization problem. It is easily understandable, so that the algorithm behavior is not obscured by too many technicalities. Our aim is not to show, which of these methods is better but rather to point out the similarities and differences.

The first problem, discused in Section 2, is the EC and ACO data structures used to solve the TSP. Therefore, different operators acting on these structures will be discussed in the next section. As the parameters play an important role in both algorithms, they are analyzed later in this work, where differences between them are emphasized (Section 4). The selection problem, of great importance in EC is also discussed for ACO. The structure of populations of individuals presented for EC and ACO is discussed as the structure of populations of the agents, called ants (Section 5). The final section describes the experiments which have been performed emphasizing similar performance of both algorithms, with regard to the solution space searched while looking for the global minimum.

Nowadays, the application of algorithms, with stochastic searching to solve optimization problems is becoming more and more popular. Such methods as EC and ACO may be included in the group of nature mimicking methods: in the case of EC – genetics [9], and in the case of ACO – the mimicry of collective insects, very primitive entities able to communicate with each other [8].

Both of these methods look at the analyzed problem differently as compared to traditional optimizing methods. They see the problem as a set of potential solutions coded in a special way. These methods use non–deterministic (probabilistic) rules of choice. EC search starts from a population of individuals. Similarly, in ACO, solutions are searched for by many individuals – agents called ants. The basic difference between EC and ACO is that in EC the starting point is fixed in stochastic way. In ACO, there is an initial population of entities performing this step during the first cycle of the algorithm (using a special form of memory and methods of communication).

2 Representation of the TSP Solution in EC and ACO

This work does not concentrate on genetic terminology and analogy with biology (especially genetics). Thus, we look at EC in a special way – from the point of view of an entire chromosome, which, as opposed to a gene, is a basic element of the problem representation and it undergoes changes according to the rules of genetics. In the case of ACO, an element of the TABU–list is the smallest unit of the problem representation. Its value may be compared with the value of a gene in the particular genetic representation of the problem (e.g. path representation).

The definition of TSP in terms of EC is comparatively easy. TSP has the natural evaluation function – the total length of tour, given the distances between cities. Having a population of tours, we can easily compare any two of them. However, what is difficult is to specify precisely the definition of tour representation. The most natural tour representation is a list of cities visited while traversing. Unfortunately, this representation is not suitable for genetic operators, especially for the crossover operator. Moreover, the binary representation of the tour is not very good for the TSP. Besides, there are three vector representations for the TSP [7], [10]: an adjacency representation, an ordinal representation and a path representation. All these representations describe the tour as a list of cities, the position of a city within the list and its coding being different. The path representation is very useful and it is similar to the representation used by ACO, where the sequence of cities representing a tour is stored directly in the TABU–list. This is because the TABU–list includes the cities visited by an ant during one cycle of the ACO algorithm.

Recapitulating the essential points, one may say, that the TSP representation is very similar in both EC and ACO. The main difference lies in the moment of creation of the first permutation of cities. In EC the initial permutation of cities, i.e. the initial population, is created at random. In ACO the initial solution is obtained after the first iteration of the algorithm, when each agent-ant remembers the visited cities in a specific order. In addition, this process is continued for each agent independently.

3 Operators and Factors Altering Current Solutions

Consider the point of execution of the algorithms in which for EC we have a population of the potential solutions represented by the sequence of numbers (assuming the path representation), and for ACO – empty agents' memories which must be filled in with some data. We make in the case of EC the special swaps or sweeps in accordance with the rules defined by genetic operators [7], [10].

Apart from the classic genetic operators like mutation, inversion and crossover, the following three crossover operators for the path representation come into existence: a partially–mapped crossover (PMX), an order crossover (OX) and a cycle crossover (CX). The PMX, proposed by Goldberg and Lingle [10], creates a new individual by choosing a tour sub–sequence from one parent and preserving the order and position of as many cities as possible from the other parent. The subsequence of the tour is selected by choosing two random cut points, which make boundaries for swapping operations. The OX [7] creates the offspring by selecting a tour subsequence from one parent and preserving the relative order of cities from the other parent. The OX crossover exploits one property of the path representation – the order of cities. The CX, proposed by Oliver [10], creates an individual in a such way that each city (and its position) comes from one of the parents. The CX operator preserves the absolute position of the elements in the path sequence. However the PMX, OX and CX operators have serious disadvantages [10].

Another promising operator is so–called Maximal Preservative Operator (MPX) introduced by Miihlenbein in 1988. It works in a similar way to the PMX operator. Genetic Edge Recombination Crossover (ER) was developed by Whitley in 1989. It is an operator which is suitable for the symmetrical TSP. The ER operator attempts to preserve the edges of the parents in order to pass on a maximum amount of information to the offspring. The breaking of edges is seen as unwanted mutation.

Grefenstette [10] proposed a class of heuristic operators that emphasizes the lengths of edges. His algorithm defines the probability distribution over selected edges based on their costs, the selection of the edges being based on this distribution. All the operators discussed above work with the path representation in TSP.

A heuristic operator proposed by Grefenstette, works in a very similar way to the ACO state transition rule. This rule called a random–proportional rule (1) describes the probability with which ant k in city r chooses city s to move to:

$$p_k(r,s) = \begin{cases} \frac{[\tau(r,s)]\cdot[\eta(r,s)]^\beta}{\sum\limits_{u\in J_k(r)}[\tau(r,u)]\cdot[\eta(r,k)]^\beta} & \text{if } s \in J_k(r) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where:

- $\tau(r,s)$ corresponds to the amount of the pheromone laid on edge (r,s),
- $\eta(r,s) = \frac{1}{d(r,s)}$ is the reciprocal of the distance $d(r,s)$,
- $J_k(r)$ is the set of cities which remain to be visited by ant k positioned at city r,
- β is a parameter which determines the relative importance of pheromone versus distance ($\beta > 0$),
- q_0 is a tunable parameter that allows exploitation/exploration procedures during the decision making process.

In the case of ACO, the role of genetic operators is taken over by the transition rule which causes alterations during the execution of the algorithm, which in turn are recorded in the TABU–list.

4 Parameters of the Algorithms

The parameters used by the algorithms under consideration may be divided into two groups:

a) related to the size of the population and the number of cycles, or generations, (quantitative parameters): for EC – the number of individuals in a population, for ACO – the number of agents–ants solving the problem, or both algorithms – the number of iterations (or populations).

b) related to the results of operations executed by the algorithms (qualitative parameters): mutation and crossover operators for EC and parameters occuring in global and local transition rules.

The values of parameters within both groups depend on the size of the problem to be solved. The parameters of ACO from the second group occur in the global and local pheromone updating rules. Once all ants have completed their tours, the pheromone trail is updated on all edges according to the formula:

$$\tau_{(r,s)}(t, t+n) = (1-\alpha) \cdot \tau_{(r,s)}(t) + \sum_{k=1}^{m} \Delta\tau_k(r,s),$$

where:

$$\Delta\tau_k(r,s) = \begin{cases} \frac{1}{L_k}, & \text{if } (r,s) \in \text{tour constructed by ant } k \\ 0, & \text{otherwise} \end{cases}$$

$0 < \alpha < 1$, L_k is the length of the tour constructed by ant k, and m is the number of ants.

The local updating rule utilize the parameter ρ:

$$\tau_{(r,s)}(t, t+1) = (1-\rho) \cdot \tau_{(r,s)}(t) + \tau_0.$$

The qualitative parameters have a decisive influence on the speed and quality of results generated by the algorithms.

5 EC and ACO Steps

Evolutionary algorithm exploits the well–known techniques in evolutionary computation class. They maintain a population of individuals $P(t)$ in iteration n. Each individual represents a potential solution, and each solution is examined to obtain its fitness value. Therefore, a new population (in iteration $n+1$) is formed by the selected, fitter individuals. Some individuals of the new population undergo transformations (alterations) caused by the genetic operators. This algorithm belongs to the group of methods called Evolutionary Computing so its structure may be described as follows [10]:

```
procedure EvolutionaryComputing; { Evolutionary Algorithm }
begin
  t := 0;              { the discrete time      }
  initialize P(t);     { the initial population }
  evaluate P(t);
  while (not TerminationCondition) do
    begin
```

```
      t := t + 1;
      select P(t) from P(t - 1);
      alter P(t);
      evaluate P(t)
    end
end;
```

The structure of an Ant Colony algorithm may be presented in the notation of Evolutionary Computing in the following way:

```
1. Initialize - place ants in the starting cities.
2. Alter the population of ants:
   a) move every ant through all cities,
      applying a state transition rule,
   b) evaporate the pheromone trail on edges
      using the local pheromone updating rule,
   c) calculate the fitness function,
      i.e.  the length of the tour found by each ant,
   d) select the best-adapted individuals (ants) within generation,
   e) update the pheromone value on appropraiate edges.
3. If the termination condition is not satisfied then repeat step 2.
4. Stop the algorithm.
```

Obviously, in this algorithm we do not accentuate evolution of the particular individuals. They neither intercross each other nor evolve in the meaning of EC. For the present, we can only show the analogy between these two methods. In both EC and ACO, the following phases are repeated: initialization, formulation a new population and selection. Selection is executed on two different levels. In ACO, with the global–best or local–best rules [3], it ends a cycle and determines whether to continue the execution of the algorithm (in EC it works in a similar way). When we think about the selection process as a choice mechanism – it occurs during a creation of the following element of a new solution (the next element in the TABU–list) and we have used a roulette wheel similar to that for EC.

6 Learning Process

Reinforcement learning (RL) is the most popular AI concept. The power of this method comes from its simplicity and some differences between RL and the traditional supervised learning paradigm. In this scheme, an agent learns from evaluative information only. The scheme of RL tries to influence maximization of reinforcement values it receives in a long period of time. This gives the possibility of forcing the behavior of the agent by rewarding it for good performance and punishing it for bad. It is possible to use the RL mechanism to learn a problem using the meta–heuristics, called ACO. The pheromone–updating rule in ACO allocates larger amounts of pheromone to shorter tours. Thus, this rule is similar to the reinforcement learning scheme in which better solutions get higher reinforcement [6]. The pheromone trail laid on the edge plays the role of a delayed reward and memory (TABU–list) for each agent that probably going to choose this edge. This constitutes an indirect form of communication and co–operation between agents, and thus it is similar to learning with delayed rewards or Q–learning – learning basing on Q-values of rewards [4], [6].

The work of EC is also based on a probabilistic choice. The main stage in EC is the selection of individuals for reproduction and creation the new population. Competitive learning (CL) (Grossberg 1976, Kohonen 1984, Rumelhart & Zipser 1986) was developed in artificial neural networks. In the case of EC, the selection mechanism added to the competitive learning algorithm to achieve the intended goal play an important role. Such an algorithm is called competitive and selective learning (CSL). In EC we observe this competitive learning. The principle of competitive learning is simple and reasonable when a population with many good adapted individuals has been obtained. The selection mechanism utilized here is based on the fitness measure [7]. Both types of algorithms exhibit more complex behavior. Rick L. Rioto from the University of Michigan observed that in EC latent learning occurs. We believe that this mechanism occurs in AA as well. The mechanism is similar to learning process among rats, which after a period of time quickly find the way out of a maze [9].

7 Experiments and Results

In addition to the analysis of the theoretical problems, the practical performance of the algorithms was examined. The computer implementations of ACO and EC using the path representation were created. In the implementation of EC two operators: the ER crossover and the MPX were applied. The task was to find a solution to the TSP instances.

The TSP problem was tested using the following sets of parameters:

	ACO:	**EC**:	
(elitist strategy)	$\tau_0 = 1.15E - 07$	$p_c = 0.95$	(MPX, ER operator)
	$\alpha = 0.1$	$p_m = 0.015$	
	$\rho = 0.01$	$N_I = n$	
	$q_0 = 0.85$	roulette wheel	

We analyzed different strategies and generative policies in ACO [2]. We found the elitist strategy the best which, what is similar to the experiments performed by Dorigo [5]. Analyzing the experimental results, we can determine the values for parameters and choose the appropriate operators for EC. One can observe the powerful influence of the choice of parameters' values, as well as the solving problem itself. These problems were analyzed in our previous works, e.g. [1], [2].

Below we present the results of tests for several TSP problems taken from the TSP-LIB95 at `www.iwr.uni-heidelberg.de`. The investigations were carried out for both algorithms independently, but we tried to achieve the best results in the same "environment". We fix the number of iterations (100 and 200) and experiments (20). Number of individuals (ants) is equal to the number of cities.

We summarize the results in Table 1 and Figure 1.

For both algorithms, the average and best solutions are comparable (Figure 1), especially in the experiment with 200 iterations. Here, the algorithms almost always attain very good solutions which are close to each other. The bigger differences among the values obtained by ACO and ET were observed for bigger instances of TSP (the minimum relative error – see Tabble 1). The analysis of the average tour lengths in consecutive

Table 1. The best results for analyzed problem in experiments with 100 and 200 iterations (20 repetitions).

problem instance	the best known solution	**ACO** (min.) rel. err.	**ACO** (aver.) rel. err.	**EC** (min.) rel. err.	**EC** (aver.) rel. err.
lin105	14379	0	0.47	0	1.62
pr136	96772	0.29	2.34	2.21	6.54
u159	42080	0	0.88	1.35	7.17
rat195	2323	0.56	1.68	11.66	16.93
kroB200	29437	0.61	3.13	4.46	11.20
ts200	126643	1.41	2.26	1.49	10.47

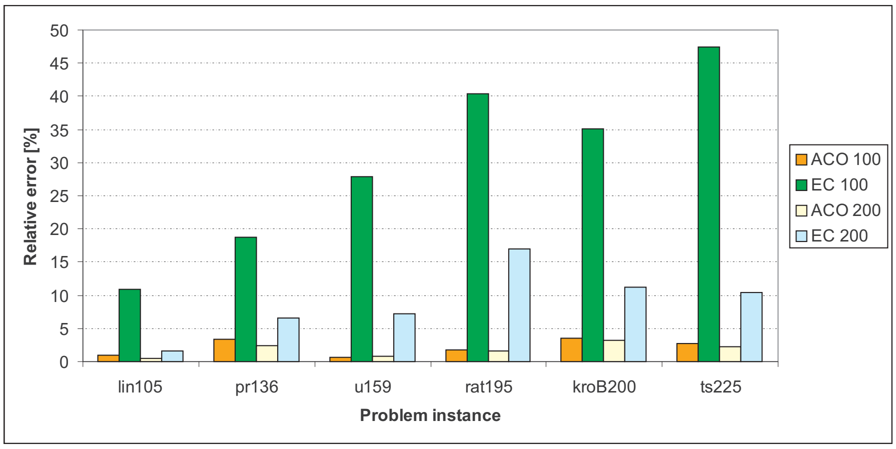

Figure 1. Comparison of ACO and EC for two experiments.

iterations shows that the Ant Colony Algorithm improves its solutions in the first 200 iterations, while the Evolutionary Algorithm give a lower performance. In the biggest instances of TSP, in both algorithms, the results are weakly different. This is a consequence of the definition of the crossover operators.The created offspring does not inherit enough adequate information from its parents.

8 Conclusions

From the experimental results showed in the previous section we can draw the following conclusions: heuristics used in ACO, especially elitism and the leader strategy, cause better performance as compared to EC. The number of agent–ants in ACO should be equal to the number of cities (the same condition refers to the number of individuals in EC). Enlarging the number of ants in ACO results in finding good solutions in shorter time. For the similar enlargement of the population GA give the solutions of the same or worse qualities. By increasing the probability of the crossover operator in EC we get better local minima but results obtained are worse than in ACO.

Comparing the algorithms, we claim that the idea of algorithm's evolution allows ACO to achieve better results in shorter time. In ACO, the co–operation among ants and synergetic effect steer the algorithm towards better solutions.

EC achieve good results too, but an unsatisfactory defined set of evolutionary operators (for altering the individuals) may result in a larger number of executed iterations. The ER crossover operator reduce the execution time of EC.

Bibliography

[1] U. Boryczka. Influence of the Problem Representation over the Selection of the Genetic Operators in the Genetic Algorithms. In *Proceedings of the Conference ,,Expert Systems"*, Wroc"law, 1997.

[2] U. Boryczka and M. Boryczka. Generative Policies in Ant System. In *Proceedings of the Conference EUFIT'97*, pages 857 – 861, Aachen, 8 – 11 wrzesie 1997.

[3] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed Optimization by Ant Colonies. In F. Vavala and P. Bourgine, editors, *Proceedings of First European Conference on Artificial Life*, pages 134 – 142, Cambridge, 1991. MIT Press.

[4] A. Colorni, M. Dorigo, and V. Maniezzo. An investigation of some properties of Ant System. In *Proceedings of the Parallel Problem Solving from Nature Conferrence (PPSN92)*, Bruksela, 1992.

[5] M. Dorigo, V. Maniezzo, and A. Colorni. Positive Feedback as a Search Strategy. Technical Report 91 – 016, Politechnico di Milano, Wochy, 1991.

[6] L. M. Gambardella and M. Dorigo. Ant–Q. A Reinforcement Learning Approach to the Traveling Salesman Problem. In *Proceedings of Twelfth International Conference on Machine Learning*, pages 252 – 260, Palo Alto, CA, 1995. Morgan Kaufman.

[7] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison–Wesley, 1989.

[8] S. Goss, R. Beckers, J. L. Deneubourg, S. Aron, and J. M. Pasteels. How Trail Laying and Trail Following can solve Foraging Problems for Ant Colonies. In R. N. Hughes, editor, *Behavioural Mechanisms for Food Selection*, volume G20, Berlin, 1990. Springer Verlag.

[9] J. Holland. *Adaptation in Natural and Artificial Systems.* MIT Press, 1975.

[10] Z. Michalewicz. *Algorytmy genetyczne + struktury danych = programy.* WNT, 1996.