Performance study of collective intelligence techniques for the Travelling Salesman Problem applied to VLSI circuits

Luna Jiménez Fernández

Escuela Técnica Superior de Ingenieros Informáticos, Universidad Politécnica de Madrid, 28660, Boadilla del Monte, Madrid
`luna.jimenez@alumnos.upm.es`

Abstract. The Travelling Salesman Problem (TSP) is a popular and important problem with several applications, including the threading of scan cells in VLSI circuits. A comparative study of important collective intelligence algorithms (Ant Swarm Optimization, Particle Swarm Optimization and Genetic Algorithms) has been performed on several instances of VLSI TSP problems of different sizes (between 131 and 343 nodes) in order to study the best tour length and time cost as the problem size increases. The results show that Ant Colony Optimization provides the best results for lower size problems, with Genetic Algorithms outperforming it in best tour length for bigger problems at the expense of its time cost.

Keywords: Ant Colony Optimization · Particle Swarm Optimization · Genetic Algorithm · Travelling Salesman Problem · VLSI.

1 Introduction

The *Travelling Salesman Problem (TSP)* is a well-known, NP-hard combinatorial optimization problem [2] which consists on finding the complete, minimal-cost tour of a graph that visits each node exactly once. Due to the exponential increase in cost (time and memory) as the problem size increases, metaheuristics and collective intelligence approaches are typical ways of solving this problem.

This algorithm can be applied to several real-life applications such as the *threading of scan chains* in Very Large Scale Integration (*VLSI*) circuits [3]. These problems all offer similar structures differing mostly in the number of scan cells. Optimal solutions to these problems could lead to chip designs with a lower chip area and a higher signal speed, therefore offering a better performance.

Thus, the goal of this paper is to study the *performance* of three popular collective intelligence algorithms (*Ant Colony Optimization*, *Particle Swarm Optimization* and *Genetic Algorithms*) on *VLSI TSP* problems as the problem size increases, and to study their performance in more detail on a big instance of the problem.

2 Models

2.1 Ant Colony Optimization

Ant Colony Optimization (ACO) algorithms are a family of population-based metaheuristics based on the use of *agents* (ants) that iteratively construct solutions by moving between nodes in a graph, guided by *pheromone trails* and *heuristic values* for each arc [6].

This algorithm can be easily applied to TSP problems, by having each ant start in a random city and randomly travel to unexplored cities based on the pheromone trail and the heuristic, building a solution as they explore the graph.

One of the main improvements proposed to the basic ACO algorithms is the *MAX-MIN Ant System* [6], which provides . In this algorithm, only the ant with the best solution found is allowed to deposit pheromones on the trail. In addition, the pheromone range is limited to an interval $[\tau_{min}, \tau_{max}]$ to avoid search stagnation. Finally, all pheromones are initialized to τ_{max} in order to incentivize exploration at the start of the algorithm.

2.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population-based metaheuristic designed to solve continuous optimization problems by using a swarm of potential solutions (particles) that move on a continuous space based on the global best solution and their individual best solution [4].

Since this approach cannot be directly applied to TSP (as it is a discrete optimization problem), a variation of the algorithm called *Discrete PSO* [4] was proposed. In it, most of the operators were redefined in order to work with TSP: *difference* between x and y is the edge exchanges necessary for y to become x, *multiplication* is selecting a portion of edge exchanges from a difference and *the computation of the velocity and the position* is based on the use of *centroids* with an added extra random velocity, in order to provide variability [4].

2.3 Genetic Algorithm

Genetic Algorithms (GA) are a family of population-based metaheuristics that mimic the *survival of the fittest* amongst species (solutions) that are generated via reproduction between previous solutions and random mutations [2].

GAs are popular for their flexibility and adaptability to each problem. Studies show that, for TSP, the best results are obtained using *rank-based selection* [5] amongst the population, using *sequential consecutive crossover* (SCX, based on choosing the best edges available from both parents) and *array index swapping* for crossover and mutation respectively, along with *elitism* [2] to construct the new generation.

3 Experimentation

3.1 Setup

General setup : Three different sized problems of VLSI TSP have been selected from the TSPWIKI [1]: *xqf131*, *xqg237* and *pma343*. The *cost* of the tours is computed as the euclidean distance of all edges in the tour.

General setup : All algorithms will run for *300* iterations a total of *5 times*. The *best tour length* and the *time taken* will be measured. A fixed *seed 0* has been used for reproducibility.

Ant Colony Optimization A *MAX-MIN Ant System* based on [6] has been implemented with *25 ants*, $\rho = 0.25$, $\alpha = 1$ and $\beta = 1$. The *pheromone upper level* is $\tau_{max} = \frac{1}{\rho} \cdot \frac{1}{T^{gb}}$ and the *pheromone lower level* is $\tau_{min} = \tau_{max}/2 \cdot n$. The heuristic used is *the inverse of the distance between nodes.*

Particle Swarm Optimization A *Discrete PSO* approach based on [4] has been implemented with *100 particles*, $\alpha = 1$ and $\beta = 1$. The operators used for *difference*, *product*, *addition* and *velocity* are the same as the improved proposal defined in [4].

Genetic Algorithm A standard *Genetic Algorithm* approach based on [2] has been implemented with *population size 200*, *crossover chance* $= 1$ and *mutation chance* $= 0.1$. A *rank-based approach* is used for selection, the *SCX operator* is used for crossover, *index swapping* is used for mutation and *elitism* is used for replacement.

3.2 Results

The results of the algorithms can be seen in Table 1. There are two main points to study: the *evolution of the algorithms* as the problem size increases and the *performance* of the algorithms on the biggest problem size (343 nodes).

Evolution with problem size The evolution of the best tour length and the cost can be seen in Figure 1. For the *average tour length*, both *ACO* and *GA* increment linearly with the problem size, providing similar tour lengths between them (both being around twice the optimal solution). *ACO* provides better results for smaller problems (131 and 237 cities), while *GA* outperforms it for bigger problem sizes (343 cities). *PSO* provides bad results, with very high tour lengths that grow exponentially with problem size, so it can be ignored.

PSO provides the best *time cost*, being considerably lower and increasing linearly. In contrast, both *ACO* and *GA* have much higher time costs that increase exponentially with size, with *GA* increasing at a notably faster rate.

Problem	Ant-Colony Optimization		Particle Swarm Optimization		Genetic Algorithm		Optimal solution	
	Length	**Time**	**Length**	**Time**	**Length**	**Time**	**Length**	**Time**
xqf131	1031.18 +/- 19.01 (987.85)	284.98 +/- 1.36 (282.91)	3328.85 +/- 43,98 (3254.31)	39.08 +/- 1.74 (37.65)	1132.10 +/- 31.14 (1102.68)	672.22 +/- 61.45 (596.93)	564	-
xqg237	2351.82- +/- 66.42 (2252.08)	1073.24- +/- 4.71 (1067.38)	9884,20 +/- 85.22 (9734.29)	105.64 +/- 4.46 (101.61)	2673.73- +/- 83.13 (2559.19)	2906.84 +/- 182.06 (2773.38)	1019	-
pma343	3933.89 +/- 144.55 (3662.97)	2717.22 +/- 41.18 (2674.78)	28009.31 +/- 305.22 (27494.62)	196.74 +/- 5.24 (190.57)	2970.51- +/- 74.19 (2878.26)	7897.12- +/- 450.29 (7562.41)	1368	-

Table 1: Results of the experimentation (300 iterations run 5 times). Each cell contains *mean value +/- standard deviation (best value)*. *Time* is measured in seconds.

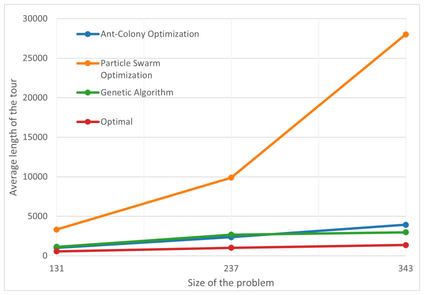
Thus, taking both parameters in consideration, *ACO* provides the better results (in tour length and time cost) for smaller sized problems, with *GA* outperforming it in tour length for bigger sized problems at the expense of a much higher time cost.

Performance at max size (343 cities) The performance of the algorithms for the biggest problem considered (343 cities) can be seen in Figure 2. The evolution of the best iteration tour and the best global tour is very similar, so both will be studied at the same time. Both *ACO* and *GA* quickly converge to a local optimum (*GA* at around 10 iterations and *ACO* at around 50), While *ACO* has some exploration in the initial iterations (with a slight increase in tour lengths before converging to a much lower one), *GA* very quickly converges into exploiting a better local optimum than *ACO*. These results suggest that a lower number of iterations could achieve the same results, thus reducing the time cost seen previously.

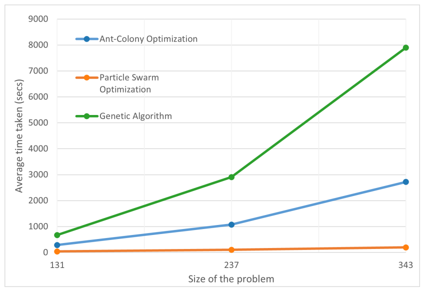
While *PSO* constantly improves during the 300 iterations, its tour lengths are several times longer than those of *ACO* and *GA*, making this algorithm effectively useless for TSP despite its very good time costs seen previously.

4 Conclusions

In this study, the performance (tour length and time cost) of several algorithms for VLSI TSP problems of different sizes has been compared by running experiments. With the proposed parameters, for smaller problems (up to around 240 cities), *Ant Colony Optimization* provides the best results both in tour length and time cost, while for bigger problems *Genetic Algorithms* provide better tour lengths at a steep time increase. *Particle Swarm Optimization* results are considerably worse for all problem sizes.

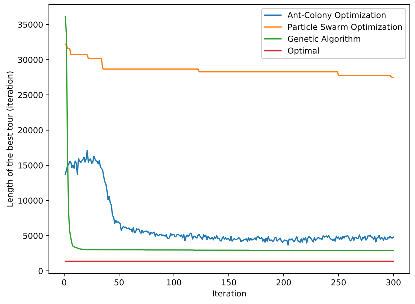


(a) Average length

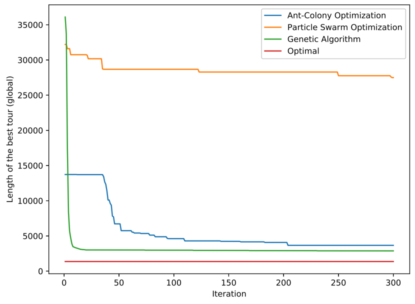


(b) Average time cost

Fig. 1: Evolution of the *algorithms' performances* when increasing problem size.



(a) Iteration best length



(b) Global best length

Fig. 2: Evolution of the algorithms' best lengths over iterations for *pma343*.

In addition, *Ant Colony Optimization* and *Genetic Algorithms* converge to local optima in few iterations. Thus, time costs can be reduced by decreasing the number of iterations.

References

1. VLSI Data Sets, http://www.math.uwaterloo.ca/tsp/vlsi/index.html
2. Ahmed, Z.H.: Improved genetic algorithms for the travelling salesman problem. International Journal of Process Management and Benchmarking (2014)
3. Boese, K., Kahng, A., Tsay, R.S.: Scan chain optimization: Heuristic and optimal solutions (01 1997)
4. Hoffmann, M., Mühlenthaler, M., Helwig, S., Wanka, R.: Discrete particle swarm optimization for TSP: theoretical results and experimental evaluations. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (2011)
5. Razali, N.M., Geraghty, J., Others: Genetic algorithm performance with different selection strategies in solving TSP. In: Proceedings of the world congress on engineering. vol. 2, pp. 1–6. International Association of Engineers Hong Kong (2011)
6. St, T., Dorigo, M.: ACO Algorithms for the Traveling Salesman Problem (1999)