

# Aircraft trajectories separation using parallel stochastic algorithms

Xavier Olive

February – July 2006

## Abstract

We propose to solve aircraft conflicts using stochastic methods, in particular ant colony algorithms, first described by Marco Dorigo in [Dorigo and Caro, 1999].

Classical optimization methods fail to solve large scale aircraft conflict problems. Nature-inspired algorithms such as genetic algorithms are particularly efficient and are commonly used to solve these problems. We will try to determine how feasible the idea of copying the behavior of ant colony trying to find the shortest path to food resources is.

We will then try different parallel methods to enhance performances of computation and to reduce convergence time.

## 1 Ant Colony Optimization (ACO)

### 1.1 Description of the algorithm on the Traveling Salesman Problem (TSP)

Given a number of cities and the costs of traveling from one city to another, what is the cheapest round-trip route that visits each city exactly once and returns to the starting point?

This problem can be represented by a weighted graph where vertices represent cities, edges represent roads and the weight of each edge is proportional to the distance between the two cities.

The main idea of ACO is to let ants try different paths, evaluate them, and put a certain amount of pheromone on each edge of the trip, proportional to the evaluation of the whole trip. The ants, attracted by pheromones, will follow high-rated paths and the algorithm will converge to the highest rated path. Figure 1 shows ants, although only having a limited knowledge of their environment, manage to communicate and select shorter ways to their destination.

For each iteration, each ant will try a particular trip. The choice of the next city will depend on  $J_i^k$ , the list of visited cities by ant  $k$  located at city  $i$ ; on  $\eta_{ij}$ , the inverse of the distance between city  $i$  and  $j$ ; and on  $\tau_{ij}$ , the amount of pheromones held by a given edge.

The probability of ant  $k$  traveling from city  $i$  to city  $j$  at instant  $t$  is:

$$p_{ij}^k(t) = \begin{cases} \frac{\tau_{ij}(t)^\alpha \cdot \eta_{ij}(t)^\beta}{\sum_{l \in J_i^k} \tau_{il}(t)^\alpha \cdot \eta_{il}(t)^\beta} & \text{if } j \in J_i^k \\ 0 & \text{if } j \notin J_i^k \end{cases}$$

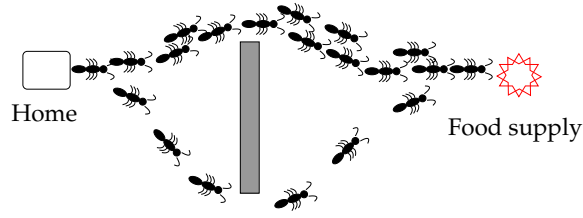


Figure 1: Ants selecting better paths

Parameters  $\alpha$  and  $\beta$  control the influence of distance and of pheromones on the choice of next city. At the end of each iteration, we spread pheromones on the edges and evaporation is simulated in order to avoid local extrema.

$$\tau_{ij}(t+1) = (1 - \rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t)$$

where  $\rho$  is the parameter that controls the evaporation.

## 1.2 Aircraft conflicts problem solved by ACO

The ACO approach was tested on a difficult problem of aircraft conflicts. Let us consider  $n$  airplanes arranged around the edge of a circle with converging trajectories. They will all be in conflict<sup>1</sup> (figure 2).

Directions are then to be given to the airplanes in order to avoid all the conflicts with a minimum diversion from the initial route.

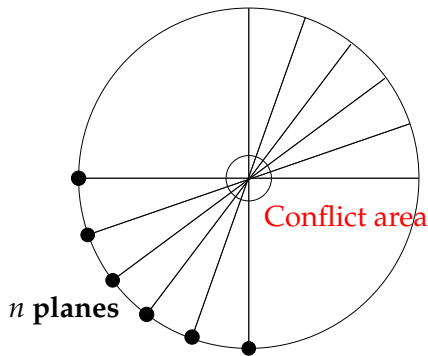


Figure 2:  $n$  planes in conflict

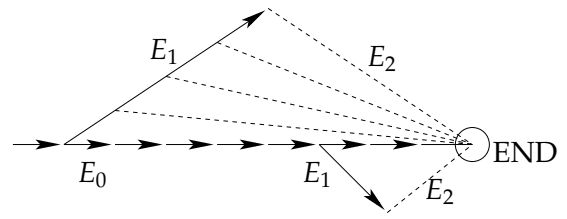


Figure 3: Choice of a path

We will associate one plane to one ant. Each ant will try to find its way through the cluster of aircraft. If no conflict appears, then each ant will spread pheromones on the path of its choice. Otherwise, no pheromone will be spread. The paths are chosen according to figure 3.

We will add 0 to global evaluation of the current path when a plane flies straight ahead ( $E_0$ ), 2 when a plane diverges ( $E_1$ ) from the main path and 1 when a plane turns back ( $E_2$ ) to his goal ( $END$ ). However, since almost no solution without conflicts are found by the

<sup>1</sup>When two airplanes are closer than 5 nautical miles, they create a conflict.

algorithm, we will first loosen the constraints: the ants will spread pheromones if less than  $k$  conflicts remain. Thus, we will be able to attract the next ants to better solutions. When a predefined number of ants is allowed to spread pheromones, the constraint is strengthened:  $k$  is replaced by  $k - 1$ . When  $k = 0$ , pheromones will be spread on paths which do not create any conflict.

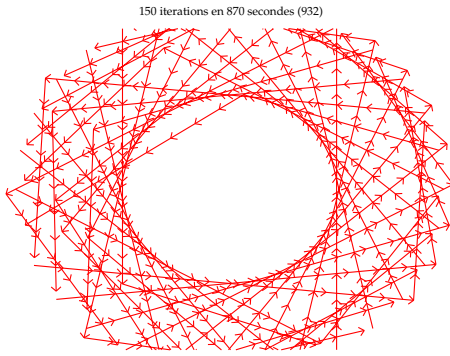


Figure 4: Solution to a 30-airplane conflict problem

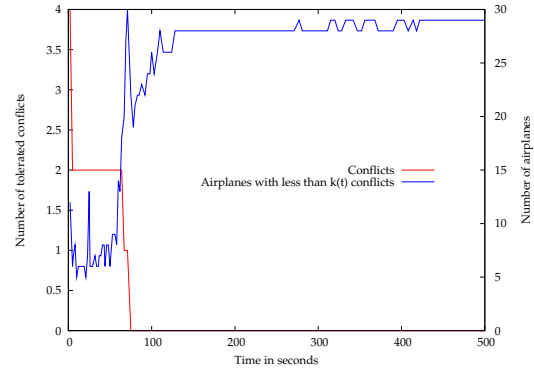


Figure 5: Convergence for a 30-airplane conflict problem

## 2 Parallelization of the algorithm

### 2.1 Master-slave design

Figure 5 shows that ACO failed to solve the 30-airplane problem. We could only find a solution with 29 airplanes.

We will now try use the benefits of parallelization to find a solution to this problem.

The first idea is to execute the time consuming operations on remote computers. These remote *slave* computers will be designed to move ants and check for conflicts. A *master* computer will gather all the information and send it back to all the slave computers, which will be able to use up-to-date pheromone data.

The design from figure 6 finds a solution to the problem. However, the main problem is that the master computer saturates quickly because of the amount of data it has to handle. Indeed, all remote computers consume almost the same time to execute their tasks and try to feed data back to the master at the same moment. By using multithread and asynchrony techniques, we can reduce convergence time by more than 2 (see figure 7).

Exact methods converge to the best solution, but are unable to solve large-scale problems; ant colony algorithms converge to some solution, but are unable to prove it will be the best one.

Therefore, we will present other designs, which will be better fitted for increasing the number of solutions among which we could search for the better ones. The current design is able to find some solutions, but we have to use a better method for an efficient optimization.

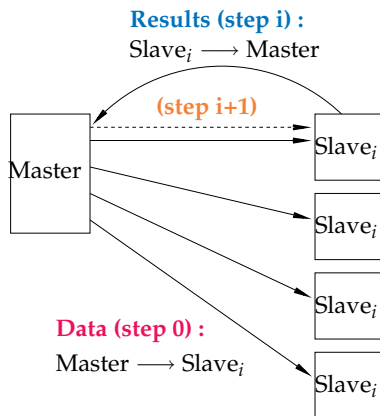


Figure 6: Master-slave design

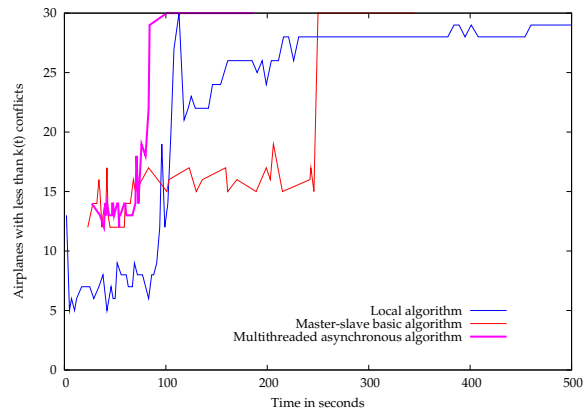


Figure 7: Comparing performance

## 2.2 Islets design

Lefablec [Lefablec, 1995] used an islets design, when trying to solve aircraft conflicts with genetic algorithms. Each islet is supposed to solve the given problem. We expect that all the islets will converge to different solutions. They exchange a small amount of information about their pheromones from time to time.

When we implement this solution for the 30-airplane problem, we realize that this solution requires each islet to converge alone. Each islet executes a local algorithm, thus if it cannot converge, the global system is unlikely to converge.

## 2.3 Blended design

We decided then to blend both master-slave and islets design: each islet will be a master-slave system, able to find its own solution, but will sometimes exchange information with the other islets. Figure 8 illustrates this design.

We expect that among all the islets, a better solution will be found. Thus we have benchmarked our design on a 12-computer cluster. On figure 9, the x-axis represents the number of islets and the y-axis represents the evaluation of the solutions found.

We displayed the evaluation of the found solutions after the algorithm is executed for 500 seconds.

This blended design was particularly efficient and matches our objectives. The quality of the found solutions is much better with 2 islets. 3 or 4 islets are not as good a compromise because there are fewer computers in the master-slave design, which is not accurate for a good convergence inside each islet.

## Conclusion

The blended design has been particularly efficient for solving aircraft conflict problems. However, this is only an ad hoc design, unfitted to solve other problems such as when the solution can be easily found inside a given domain. For instance, when we try to parallelize the Traveling Salesman Problem, results are found to be better with the basic islet design.

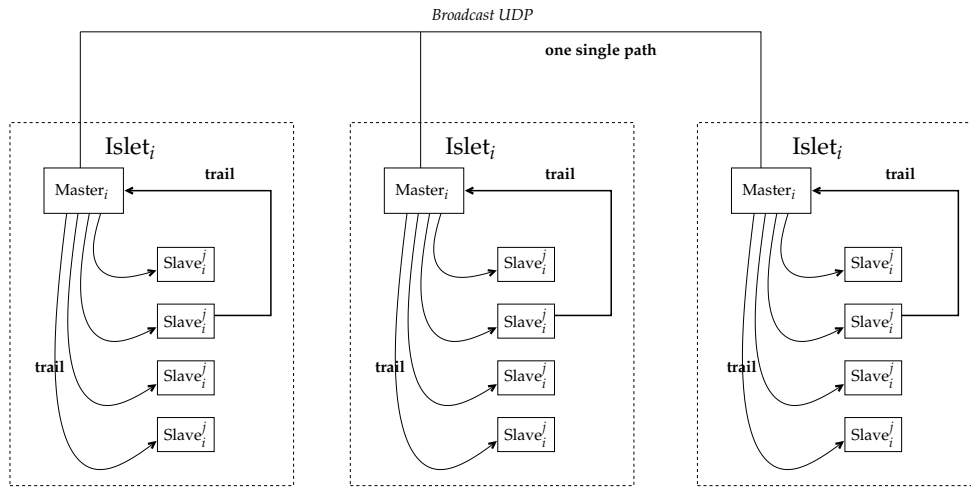


Figure 8: Blended design

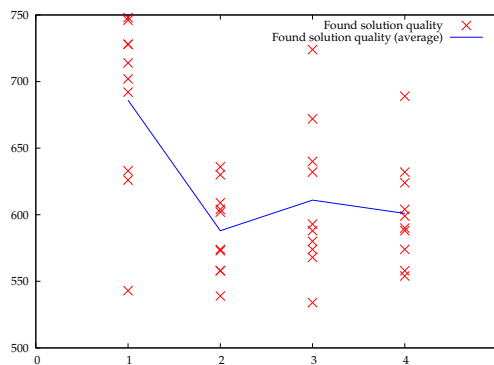


Figure 9: Performances

Up til now, the ACO has been found to be less efficient for all problems in comparison to any other metaheuristics. However, it was important that we compare the ACO and the genetic algorithm in terms of their performances in resolving conflicts.

## References

- [Dorigo and Caro, 1999] Dorigo, M. and Caro, G. D. (1999). Ant Colony Optimization: a New Meta-Heuristic.
- [Lefablec, 1995] Lefablec, Y. (1995). Optimisation par algorithmes génétiques parallèles et multi-objectifs. Master's thesis, Rapport DEA IFP.