Meta-Heuristic Algorithm based on Ant Colony Optimization Algorithm and Project Scheduling Problem (PSP) for the Traveling Salesman Problem

Fuentes-Penna Alejandro
Escuela Superior de Tlahuelilpan – Universidad Autónoma del Estado de Hidalgo (ESTl – UAEM)
alexfp10@hotmail.com

Ruiz-Vanoye, Jorge A.
Universidad Autónoma de Ciudad del Carmen.

González-Ramírez Marcos S.
Universidad Autónoma del Estado de Morelos

Abstract
The main target of Traveling Salesman Problem (TSP) is to construct the path with the lowest time between different cities, visiting every one once. The Scheduling Project Ant Colony Optimization (SPANCO) Algorithm proposes a way to solve TSP problems adding three aspects: time, cost effort and scope, where the scope is the number of cities, the effort is calculated multiplying time, distance and delivering weight factors and dividing by the sum of them and optimizing the best way to visit the cities graph.

Keywords: Project Scheduling Problem, Ant Colony Optimization, Traveling Salesman Problem, Scheduling Project Ant Colony Optimization.

1 Introduction

1.1 Nature simulation
In a colony of social insects, such as ants, bees, wasps and termites, each kind of insect performs its own tasks independently from other members of the colony but with complementary activities; this tasks performed by different insects are related to each other in such a way that the colony, as a whole, is capable of solving complex problems through cooperation [2]. The group activities as selecting and picking up materials, finding and storing food, which require sophisticated planning, are solved by insect colonies without any kind of supervisor or centralized controller. This collective behavior which emerges from a group of social insects has been called “swarm intelligence” [2].

Real ants tracks their food exploiting pheromone information, but the principal issue is how they can find the shortest path from a food source to their nest [15], [13] without using visual cues [14].

2. Project Scheduling Problem
Every project has a set of activities involving a target and a set of restrictions. Each activity has a set of restricted resources and an assigned cost; in this scenario, Project Scheduling Problem (PSP) is a way to solve this kind of problems where it can be considered as a generic name given to a whole class of problems in which it is necessary scheduling of optimum way the time, cost and resources of projects [3].


Ruiz-Vanoye et al [6] describe the project management as “the application of knowledge, abilities, tools and techniques to activities of projects so that they fulfill or exceed the needs and expectations of a project, such as: reach, time, cost and quality, requirements identified (needs) and requirements non-identified (expectations)”.
There are many types of problems where PSP can be applied:

- Technical elements (development of software, pharmaceutical drugs or civil engineering, Production systems planning) [6].
- Elements of the administration (Project Scheduling Problems, Manufacturing Management, Technology Management, contracts with the government or development of new products) [6].
- Groups of industry (industrial engineering, automobiles, chemicals or financial services) [6].
- Bio - inspiration elements to choose the best option.
- Biotechnology elements to choose the best way to obtain the best elements combination.
- Scholar elements (Timetabling Scheduling, scholar administration)
- Health elements (nursing process, drug administration, patient care management)

Huang et al, 2009 [4] defines a software project as a directed acyclic graph G = (V, A, S, E) where V = (1, 2, …, n) is the set of nodes representing the events (figure 1). A is the set of arcs representing the activites, (i, j) ∈ A is the arc acyclic graph G from node i to j, and there is only one directed arc (i, j) from i to j, S ∈ V is the start node, and E ∈ V is the end node. Each activity duration time be a stochastic variable denoted ξ = {ξij (i, j) ∈ A} . The capital cost of activity through (i, j) is denoted Cij while the resource cost of activity through (i, j) is denoted r_{ij}.

Ruiz-Vanoye et al [23] classify the main variants of PSP:

- Resource-constrained Project Scheduling Problem (RCPS),
- Multi-Mode Resource-Constrained Project Scheduling Problem (MRCPS),
- Construction Project Scheduling Problem (construction PSP),
- Project Scheduling Problem for Software Development (software PSP),
- Payment Scheduling Problem,
- Time/cost trade-off Problem (TCTP).

Fuentes et al [24] proposed PSP with Linear Responsibility Chart (LRC) to generate an initial workload and a project initial total cost.

3 Traveling Salesman Problem

The traveling salesman problem (TSP) [10] models the situation of a travelling salesman who is required to pass through a number of cities without visiting twice the same city, so that the total travelling distances be minimal (Figure 3). According to Papadimitriou and Steiglitz [11], a Combinational Optimization Problem (COP) P = (S, f) is an optimization problem in which are given a finite set of objects S and an objective function – maximize or minimize – f :S →R+ that assigns a positive cost value to each of the objects s ∈ S. The objects are typically integer positive numbers, subsets of a set of items, permutations of a set of items, or graph structures. COP can be modelled as discrete optimization problems in which the search space is defined over a set of decision variables Xi i = 1, …, n, with discrete domains. Therefore, we will henceforth use the terms CO problem and discrete optimization problem interchangeably.

Zambito [31] describes the TSP as a graph G with a set of vertices V, a set of edges, E, and a cost, c_{ij}, associated with each edge in E; the author mentions the solution to the TSP must return the cheapest Hamiltonian cycle of G.
At [10], the author refers the TSP is a set of n towns, where the TSP can be stated as the problem of finding a minimal length tour to visit each town once. \(d_{ij}\) is the length of the path between towns i and j; in the case of Euclidean TSP, \(d_{ij}\) is the Euclidean distance between i and j. An instance of the TSP is given by a graph \((N,E)\), where \(N\) is the set of towns and \(E\) is the set of edges between towns. \(b_i(t)\) \((i=1, ..., n)\) is the number of ants in town i at time t and \(m\) is the total number of ants.

Goyal [25] Classified TSP on different instances:

- Symmetric TSP
- Asymmetric TSP

The deterministic solutions to TSP are:

- Linear programming [26].
- Dynamic programming formulation for an exact solution of TSP [27].
- The branch and bound technique [28].
- Graphs with degree at most 3 in exponential time [29].

The non-deterministic solutions to TSP are:

- Nearest neighbor algorithm [30].
- Insertion algorithms [30].
- K-Opt Heuristics [30].
- Greedy Non-Deterministic Solution [25]

There are many practical real life uses of the TSP, where each node can be presented as a city, geographical location, buildings, locations, electronic points, etc.; the most common of which are [31]:

- Transportation routing problems
- Salespersons
- Music band on tour

Postal delivery person

4 Ant Colony Optimization (ACO)

The first ACO algorithms are presented by Marco Dorigo et al at the early 1990's [1]. The development of these algorithms was inspired by the observation of ant colonies. The behavior that provided the inspiration for ACO is the ants searching for food, ants initially explore the area surrounding their nest in a random manner (Figure 2). While moving, ants leave a chemical pheromone trail on the ground. Ants can smell pheromone. When choosing their way, they tend to choose, in probability, paths marked by strong pheromone concentrations. As soon as an ant finds a food source, it evaluates the quantity and the quality of the food and carries some of it back to the nest. During the return trip, the quantity of pheromone that an ant leaves on the ground may depend on the quantity and quality of the food [11].

Dorigo et al [21] present the General ACO meta-heuristic:

```
procedure ACO metaheuristics
    ScheduleActivities
        ManageAntActivity()
        EvaporatePheromone() // forgetting
        DaemonActions() {optional} // centralized actions
    end ScheduleActivities
end ACO metaheuristics
```

Steps for implementing ACO [21]:

- Choose appropriate graph representation
- Define positive feedback
- Choose constructive heuristic
- Choose a model for constraint handling (tabu list at TSP)

4.1 Algorithms to solve TSP with ACO

Dorigo [11] proposes a cooperative learning approach based on ant colony system where a set of cooperating agents called ants cooperate to find good solutions to TSP's.

Stutzle and Hoos [16] solved the TSP based on pheromones trail and the heuristic information between nodes choosing between the maximum and the
minimum trail strengths and constructing a valid tour according to a probability distribution proportional. They proposed a Max−Min ant System where they select the best ant to update the pheromones trail.


Junjie and Dingwei [19] used an ACO Algorithm for multiple TSP, where the total salesmen visited every unvisited nodes selecting the next node independently.

Salami [20] proposed the ACO Genetic Algorithm (ACOG) adapting genetic operations to enhance ant movement towards solution state.

The way ant colonies function has suggested the definition of a computational paradigm, which we call Ant System This solution can apply to TSP because the solution consists on visit stages evading obstacles. Dorigo et al [11].

Dorigo and Di Caro [17] mentions: “(i) it is relatively easy to adapt the ant colony metaphor to it, (ii) it is a very difficult problem (NP-hard), (iii) it is one of the most studied problems in combinatorial optimization (Lawler et al., 1985; Reinelt, 1994), and (iv) it is very easy to state and explain it, so that the algorithm behavior is not obscured by too many technicalities.”

5. Proposed a Scheduling Project Ant Colony Optimization (SPANCO) Algorithm

A combinatorial optimization problem has a set C = {c₁, …, cₘ} of components. A subset cᵢ of components represents an optimal solution of the problem; There are a 2ᶜ possible solutions combination (S) and the selected solution Sᵢ is feasible if Sᵢ ∈ S. A time function t is defined over the solution domain and the objective is to find a minimum time solution S₁. The ants move is based on stochastic local decision policy with two parameters: local pheromone update rule, which applied whilst constructing solutions and global pheromone updating rule, which applied after all ants construct a solution.

The proposed algorithm is combined well distribution strategy of initial ACO and PSP for assign the best way. The recognition group explores the different paths according to ACO to visit every node.

The main loop consists on Estimate the effort’s cost between nodes following the pheromone trail, select the next node with minimal effort cost (the node with minor path, max pheromones and minor distance), subtract the delivered package weight of the total weight and update the visited nodes.

The SPANCO main target is determine the best way (optimization loop) to travel between nodes selecting the next node based on the recognition loop and the main loop.

The proposed algorithm is described as follows:

```plaintext
//Initialization
Set parameters, initialize pheromone trails
m = nodes (cities)
costᵢⱼ = Travel cost between nodeᵢ and nodeⱼ
Wᵣ = Total package’s weight to deliver
Wⱼ = package’s weight delivered on nodeⱼ
distanceᵢⱼ = Distance between nodeᵢ and nodeⱼ
N = Total of ants
k = random number of ants N
\text{ant}_{p} = \text{actual ant}

// Functions
Update Pheromones()
Total_Pheromonesᵢⱼ =
Total_Pheromonesᵢⱼ +
\text{Pheromones}_{ant_{p}} -
\text{random_diffusion_pheromones}
End Update

max_Pheromones()
- Compare k - pheromones from pathsₚ and select the pathₚ with max pheromones
End max_pheromones

min_pathₚ()
- Compare k - pathsₚ and select the pathₚ with min total distance
End min_path

Estimate cost_effortᵢⱼ(timeᵢⱼ, distanceᵢⱼ, Wᵣ)
```

387
If \((\text{time}_{ij} > 0, \text{distance}_{ij} > 0, W_t > 0)\)

\[
\text{cost\_effort}_{ij} = \frac{\text{time}_{ij} \times \text{distance}_{ij} \times W_t}{\text{time}_{ij} + \text{distance}_{ij} + W_t}
\]

End if

End Estimate

Update Weight delivered()

\[W_t = W_t - W_j\]

End Update

Trace_best_way() selecting node\(_j\) to visit where \((\max\_pheromone() \text{ and } \min\_cost\_effort_{ij}() \text{ and } \min\_path_{ij}())\)

Select neighbor nodes from node\(_i\)

Compare neighbor – paths\(_p\)

(pheromone\(_{ij}()\), cost\_effort\(_{ij}()\), path\(_{ij}())\)

Select the best way with

\((\max\_pheromone_{ij}(), \min\_cost\_effort_{ij}(), \min\_path_{ij}())\)

End trace_best_way

// Recognition loop

Allocate \(h = \text{random number of} \ k \text{ on initial (or random) node}_i \text{ where } i \leq m\)

for each ant\(_p\) from \(k\) group

Select random node\(_j\) to visit

Do

Estimate \text{time}_{ij}

Estimate \text{distance}_{ij}

Update nodes visited

Update pheromones()

Assign \(\text{path}_p = \text{path}_p + \text{distance}_{ij}\)

node\(_i\) = node\(_j\)

Select random node\(_j\) to visit

While node\(_j\) was not visited

end for

//Main loop \(n=N-k\)

For each ant from \(n\) group

// second ants group

Allocate \(n = N-k\) ants on initial (or random) node\(_i\) where \(i \leq n\)

End For

Do

Estimate \text{cost\_effort}_{ij}(\text{time}, \text{distance}, \text{Actual\_weight}) \text{ from neighbor nodes}

Select node\(_i\) to visit where

\((\max\_pheromone() \text{ and } \min\_path_p() \text{ and } \min\_cost\_effort_{ij}())\)

If \(\max\_pheromone() = 0\)

Select random node,

End if

Deliver Weight\(_i\)()

Update Weight delivered()

Update cities visited()

Update pheromones()

\(i=j\)

While not \_visited\_all\_nodes()

//Optimization loop

initial (or random) node\(_i\) // the origin city

do

trace_best_way() selecting node\(_j\) to visit where

\((\max\_pheromone_{ij}() \text{ and } \min\_cost\_effort_{ij}() \text{ and } \min\_path_{ij}())\)

\(i=j\)

While not \_visited\_all\_nodes()

5.1 SPANCO Algorithm

An SPANCO algorithm differs from the algorithms proposed; it uses a Project Scheduling Problem (PSP) to enhance performance. It consists of three main sections: initialization, main loop and optimization loop. The main loop runs for a user defined number of iterations. These are described below:

Initialization

a. Set the initial parameters: variables, initial states, functions, inputs, outputs, initial trajectory, output trajectories.

b. Initial set of nodes (cities)

c. Set initial pheromone trails value = 0.

d. Set the travel costs between nodes

e. Estimate the total weight of deliver packages

f. Estimate each weight’s package to deliver on each node

g. Define the total ants

h. Select an ants random number from total ants

i. Set the distances between nodes

j. Each node has empty pheromones.

Functions

a. Update pheromones(): Ants leave a pheromone trail which is updated when
another ant goes through the same path and randomly the trail is removing for natural conditions.

b. The max pheromones function compares the paths between the actual node with other nodes and select the path where more ants.

c. Min_path(): This function aims to compare the different paths used by ants recognition group and select the minimal distance path.

d. Cost_effort: Each path between two cities laid out by the ants has a time, distance and weight carried. The effort is calculated from these factors multiplying them and dividing the result by the sum.

e. Weight delivered(): if we assume that the traveling salesman leaves a package on each visited node, the total weight decreases with these deliveries.

f. Trace_best_way(): This function analyzes the different path efforts between two cities, from this analysis, we choose the path of least distance, less effort and maximum pheromone.

**Recognition loop**

a. A random ants number is allocated on initial node (or random node) to initialize the paths search (recognition group)

b. Each ant will visit every node randomly estimating the time and distance between nodes and delivering pheromones

**Main loop**

a. Set the initial parameters with the results of initialization and recognition loop.

b. Each ant is individually placed on initial state with empty memory.

c. The rest of the ants colony is allocated on the same initial node (or the same random node) from recognition group

d. Set initial pheromone trails value based on recognition group.

e. Each visited city has ant’s pheromones from recognition group.

f. Construct Ant Solution while each ant has not visited every city: Each ant constructs a path by applying the transition function between node (i,j) based on the data from recognition group and the new data from actual ant; the probability of moving from node to node depend on the parameters described before.

   a. Apply Local Search

   b. Estimate the cost effort from actual node to neighbor nodes based on time and distance estimated by recognition group

   c. Select the next node to visit based on minimize the cost effort, maximize the pheromones and minimize the path’s distance

   d. Update Trails: Evaporate a fixed proportion of the pheromone on each road. For each ant perform the “ant-cycle” pheromone update. Reinforce the best tour with a set number of “elitist ants” performing the “ant-cycle”.

   e. Update the weight to transport eliminating the delivered weight.

   f. Update the pheromones where the actual ant has passed.

   g. The operations are applied to individual(s) selected from the population with a probability based on fitness. Darwinian Reproduction, Structure-Preserving Crossover or Structure-Preserving Mutation.

   g. End While

**Optimization loop**

a. Set the initial parameters with the results of initialization, recognition loop and main loop.

b. Set initial pheromone trails value over initialization, recognition loop and main loop.

c. Each city has a pheromones level from N group.

d. Construct the best Ant Solution: Each ant constructed a path by applying the transition function between node (i,j).

e. the best way is calculated by multiplying the distance, time and weight and divided by the sum of these parameters:

   a. While not visited every city

      i. max_pheromone(): Analyze the paths (i,j) maximizing the pheromones level

      ii. min_cost_effort(): Analyze the paths (i,j) minimizing the cost_effort

      iii. min_path_(): select the path (i,j) minimizing the distance

   b. End While

5.2. SPANCO Simulation
The 100 cities tour simulation [22] to be solved with ACO and SPANCO to compare the solutions.

// Initialization loop:

\[ m = 100 \text{ nodes} \]
\[ \text{cost}_ij = \text{random function between 0…1} \]
\[ W_t = \text{sum of random weight package delivered on node}_j \]
\[ W_j = \text{random weight package delivered on node}_j \]
\[ \text{distance}_ij = \text{Random distance function between nodes} \]
\[ N = \text{Random Total ants} \]
\[ k = \text{Random number of ants} < N \]
\[ \text{time}_ij = 0.001 \text{ seconds} \times \text{distance} \times 1000 \]

At the initial tours we presented 100 cities to visit (figure 4). Each city is represented by a node with a random distance.

The TSP can be solved applying an aleatorial solution (AS), where the solutions present crossing ways, a mayor path distances and a mayor effort cost.

The ACO Solution (Figura 5) does not present crossing ways and the results are better than AS; the time solutions increase almost 300% comparing to AS, but the solution reduced the 20% distance approximately.

The effort cost is widely better if the AS effort cost is 100%, ACO presents de 85% averaged.

The SPANCO solution (Figure 7) used less memory than AS and ACO, the time solution is similar to ACO, but the effort cost is 20% averaged lowest than AS and in some cases, presents a 13% lowest from ACO.

5.3 Comparing solutions

The table 1 shows the comparison between AS, ACO and SPANCO results from one instance. This instance has a hundred nodes randomly distributed, ten neighbor nodes maximum, thirty-three ants from recognition group and a random weight per node.

<table>
<thead>
<tr>
<th></th>
<th>AS</th>
<th>ACO</th>
<th>SPANCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory used</td>
<td>27.5 Mb</td>
<td>27.6 Mb</td>
<td>27.5 Mb</td>
</tr>
</tbody>
</table>
### Table 1. Comparative table

<table>
<thead>
<tr>
<th></th>
<th>0.5 Sec</th>
<th>1.5 Sec</th>
<th>1.48 Sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time solution</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node swaps</td>
<td>NA</td>
<td>23 nodes</td>
<td>18 nodes</td>
</tr>
<tr>
<td>Crossing ways</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total cost</td>
<td>NA</td>
<td>100%</td>
<td>98.2%</td>
</tr>
<tr>
<td>Total distance</td>
<td>91.647</td>
<td>77.938</td>
<td>76.582</td>
</tr>
<tr>
<td>Max Neighbors</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Max recognition ants</td>
<td>33%</td>
<td>33%</td>
<td>33%</td>
</tr>
</tbody>
</table>

Table 2. 30 effort per instance comparative table

<table>
<thead>
<tr>
<th>Instance</th>
<th>ACO</th>
<th>SPANCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>85.04%</td>
<td>83.56%</td>
</tr>
<tr>
<td>2</td>
<td>98.66%</td>
<td>83.56%</td>
</tr>
<tr>
<td>3</td>
<td>97.52%</td>
<td>81.07%</td>
</tr>
<tr>
<td>4</td>
<td>97.76%</td>
<td>89.95%</td>
</tr>
<tr>
<td>5</td>
<td>88.19%</td>
<td>82.60%</td>
</tr>
<tr>
<td>6</td>
<td>92.88%</td>
<td>86.64%</td>
</tr>
<tr>
<td>7</td>
<td>94.36%</td>
<td>83.56%</td>
</tr>
<tr>
<td>8</td>
<td>93.33%</td>
<td>82.97%</td>
</tr>
<tr>
<td>9</td>
<td>90.45%</td>
<td>81.07%</td>
</tr>
<tr>
<td>10</td>
<td>93.32%</td>
<td>89.95%</td>
</tr>
<tr>
<td>11</td>
<td>87.80%</td>
<td>82.60%</td>
</tr>
<tr>
<td>12</td>
<td>89.91%</td>
<td>86.64%</td>
</tr>
<tr>
<td>13</td>
<td>94.25%</td>
<td>87.05%</td>
</tr>
<tr>
<td>14</td>
<td>88.23%</td>
<td>86.54%</td>
</tr>
<tr>
<td>15</td>
<td>91.21%</td>
<td>85.25%</td>
</tr>
<tr>
<td>16</td>
<td>88.04%</td>
<td>85.80%</td>
</tr>
<tr>
<td>17</td>
<td>93.15%</td>
<td>85.12%</td>
</tr>
<tr>
<td>18</td>
<td>86.22%</td>
<td>84.96%</td>
</tr>
<tr>
<td>19</td>
<td>87.96%</td>
<td>82.76%</td>
</tr>
<tr>
<td>20</td>
<td>91.17%</td>
<td>83.72%</td>
</tr>
<tr>
<td>21</td>
<td>83.25%</td>
<td>80.03%</td>
</tr>
<tr>
<td>22</td>
<td>92.77%</td>
<td>88.24%</td>
</tr>
<tr>
<td>23</td>
<td>85.93%</td>
<td>82.16%</td>
</tr>
<tr>
<td>24</td>
<td>84.64%</td>
<td>82.65%</td>
</tr>
<tr>
<td>25</td>
<td>97.22%</td>
<td>92.41%</td>
</tr>
<tr>
<td>26</td>
<td>90.33%</td>
<td>86.25%</td>
</tr>
<tr>
<td>27</td>
<td>88.23%</td>
<td>80.70%</td>
</tr>
</tbody>
</table>

### 6. Conclusions

Ruiz-Vanoye et al. describes the complexity of algorithms as the way of classify the algorithms about the optimal solutions on time and efficiently and the complexity of instances as the problem complex [9]; in this way the SPANCO algorithm involves a non probabilistic steps because it based on select the best way maximizing the resources and minimizing the efforts.

The AS solution has several problems to solve TSP, the Ant colony Optimization (ACO) algorithm may produce redundant states in earlier steps, but if we combine ACO with Project Scheduling Problem (PSP) adding a cost_effort function, the algorithm is better minimizing the time and effort’s costs to enhance the behavior of the inducted system.

A colony of ants moves through system states randomly based on past pheromones trails but this way not always is the best solution. If we apply PSP, the ants moves are focused on reduce the redundant states by selecting the optimum way to solve the problem.

On simulation results SPANCO has a better performance selecting the best way based on pheromones trail, the minimal path, the minimal time and these parameters benefits minimizing the carrying of delivered packages.

### 7. Related works

This is the first work introducing PSP with ACO and TSP, in future works the researches can improve the effort_cost function, introduce new paradigms or new meta heuristics. This solution can be applied on several problems with similar requirements.

### 8. Bibliography


