Applying Opposition-Based Ideas to the Ant Colony System

Alice R. Malisia, Hamid R. Tizhoosh
Department of Systems Design Engineering, University of Waterloo, ON, Canada
armalisi@uwaterloo.ca, tizhoosh@uwaterloo.ca

Abstract— **This paper presents several extensions to an algorithm in the family of Ant Colony Optimization, the Ant Colony System. The proposed extensions are based on the idea of opposition and attempt to increase the exploration efficiency of the solution space. The modifications focus on the solution construction phase of the ant colony system. Three of the proposed methods work by pairing the ants and synchronizing their path selection. The two other approaches modify the decisions of the ants by using an opposite-pheromone content. Results on the application of these algorithms on Travelling Salesman Problem instances demonstrate that the concept of opposition is not easily applied to the ant algorithm. Only one of the pheromone-based methods showed performance improvements that were statistically significant. The quality of the solutions increased and more optimal solutions were found. The other extensions showed no clear improvement. Further work must be conducted to explore the successful pheromone-based approach, as well as to determine if opposition should be applied to a different phase of the algorithm.**

I. Introduction

The concept of opposition-based learning (OBL) was recently proposed to extend different machine learning algorithms [17]. The main idea of OBL is to consider opposite estimates, actions or states as an attempt to increase the coverage of the solution space and to reduce exploration time. This should lead to increased accuracy and shorter computation time. OBL involves a general strategy that can be tailored to the technique of interest. Opposition-based reinforcement learning approaches, where opposite states and opposite actions are concurrently updated, lead to superior performance than classical reinforcement learning [16], [14]. Opposition-based extensions of neural networks [18] and the differential evolution algorithm [10] also lead to improvements in performance.

Like other machine intelligence methods, Ant Colony Optimization (ACO) algorithms are based on a phenomenon occurring in nature, the social behaviour of ant colonies [6]. Ants are well known for their ability to efficiently find the shortest path between their nest and their food source. Their incredible optimizing capacity is achieved by their ability to communicate indirectly by means of pheromone deposits [1]. ACO implementations have been successfully applied to many complex optimization problems, such as the travelling salesman problem (TSP), the quadratic assignment problem (QAP), vehicle routing, etc [6].

Despite being a powerful algorithm, ACO can remain trapped in local optima. This situation can occur when a certain component is very desirable on its own, but leads to a sub-optimal solution when combined with other components. ACO implementations are based on positive reinforcement of good solutions. Thus, after a certain number of iterations the ants will tend to select similar paths.

Since the introduction of ACO, researchers have developed multiple versions to improve the performance of the algorithm. Ant Colony System (ACS) is a commonly used extension of the original ant algorithm [6]. The ACS algorithm has a greedy selection rule, but provides online pheromone reduction (see Eq. (3)) as a measure to decrease desirability of arcs once they are travelled [5]. This prevents all the ants in the colony from generating the same solution.

Work has been conducted to establish more complex pheromone mechanisms, such as multiple pheromone matrices, complex pheromone updates, etc. These modifications were implemented so ACO could solve more complex problems and to improve the performance of ACS. For instance, one particular variant of the ant algorithm known as the best-worst Ant System (BWAS) [2] substracts pheromone content based on the results of the worst ant of the colony. It also uses a form of pheromone mutation based on concepts from evolutionary computation. To solve a bi-criterion vehicle routing problem, Iredi, Merkle and Middendorf [8] proposed a version of ACS where two different pheromone trail matrices and two heuristic functions are considered simultaneously. Schoonderwoerd et al. [13] were one of the first to elude to the concept of an ‘anti-pheromone’, where ants would decrease pheromone contents rather than reinforce them. Randall and Montgomery [11] proposed the Accumulated Experience Ant Colony (AEAC) as a method to determine the effect of each component on the overall solution quality. In their approach, the pheromone and heuristic values of an edge are weighted.

Similarly, Montgomery and Randall [9] developped three methods based ont the concept of anti-pheromone as an attempt to capture complex pheromone behaviour. In the first, the pheromone of the elements composing the worst solutions is reduced. Their second alternative combines a pheromone content for the best solution and pheromone content for the worse solution (anti-pheromone). The ants select edges based on a weighted combination of pheromone and anti-pheromone and the heuristic. Finally, their third approach involves the use of a small number of *explorer ants* that have a reversed preference for the pheromone. Their approaches produced better solutions on smaller TSP problems. Some of the extensions presented in this paper make use of a form of anti-pheromone similar to the *explorer ants*, but involve a different methodology.

Therefore, the present study is motivated by the idea of developing more complex pheromone or path selection behaviour for the ACS. This paper proposes five different extensions to the normal ACS based on the concept of opposition. The goal is to assess the validity of theses extensions, as well as establish the applicability of opposition to ACS.

The remaining of this paper is organized as follows. Section II gives an overview of the ACS, specifically applied to TSP. Section III presents the opposition-based extensions to ACS. Experimental results are included in section IV and conclusions are presented in section V.

II. Ant Colony System

The Ant Colony System (ACS) falls under the set of Ant Colony Optimization (ACO) algorithms, which are inspired from the natural behaviour of ants. ACO was introduced in 1992 by Marco Dorigo [3]. The original intent of ACO was to solve the travelling salesman problem (TSP). The first instance of ACO was the Ant System (AS) [4]. Today, the AS has been subject to many modifications to improve results. One such variation is the Ant Colony System (ACS) [5]. This version demonstrated considerable improvement over the AS.

The ACS and other ACO algorithms are based on the trail laying and following behaviour of ants. In nature, ants are easily able to find the shortest path between their nest and a food source. This is possible because they communicate with each other via pheromone deposits which are left behind as they travel. The presence of pheromone on a specific path influences its selection by the ants.

This paper presents algorithms that extend ACS. Thus, this section will provide a general description of the ACS algorithm and its governing equations specifically applied to TSP. The goal of a TSP optimization is to find the shortest path connecting a specified number of cities. The ACS algorithm works with a colony of ants that move through the network of cities seeking the optimal path.

The ants are initially distributed randomly among the cities. The number of ants, m, is usually smaller than the number of cities, n, $(m < n)$. Each ant will in turn add a component to its partial solution. The ant selects the next city based on pheromone and heuristic value. At each step of the construction, an ant k located on city i selects the next city j using the pseudorandom rule described by

$$j = \begin{cases} \underset{l \in \mathrm{N}_i^k}{\operatorname{argmax}} \left\{ \tau_{il} [\eta_{il}]^{\beta} \right\} & \text{if } q < q_o, \\ J & \text{otherwise}, \end{cases} \tag{1}$$

where τ_{il} and η_{il} represent the pheromone content and heuristic information on the edge connecting city i to city l respectively. The city l is a node that is included in N_i^k, the neighbourhood for ant k given its current location i. This neighbourhood only includes cities that have not been visited. The parameber q is a uniform random number and q_o is the probability that an ant will use learned knowledge. Thus, if $q < q_o$, the ant will select the node with the highest combined value of pheromone content and heuristic function. Otherwise, it will use J, which represents a random variable selected using a probabilistic action rule that dictates the probability for ant k to choose to go to next node j given that it is currently on node i:

$$p_{ij}^k = \frac{[\tau_{ij}][\eta_{ij}]^{\beta}}{\sum_{l \in \mathrm{N}_i^k} [\tau_{il}][\eta_{il}]^{\beta}} \qquad \text{if } j \in N_i^k, \tag{2}$$

where β represents the influence of the heuristic, which is a measure of the cost of adding the particular edge to the partial solution. The pseudorandom rule (see Eq. 1) is a greedy selection approach, that will tend to favour the edges (paths) with the best combination of pheromone and short length.

Every time an ant adds an edge to the path, the amount of pheromone on the edge is decreased using the following equation:

$$\tau_{ij}^{new} = (1 - \xi)\tau_{ij}^{current} + \xi\tau_o \quad 0 < \xi < 1, \tag{3}$$

where τ_o is the initial amount of pheromone and ξ is the local evaporation rate. This local update works to counterbalance the greedy construction rule by reducing the pheromone on the selected edge, thus making it less desirable to the next ant.

Finally, when all the ants have completed their paths, the solutions are evaluated. If a better solution was found, the best solution achieved so far by the algorithm is updated. Then, a global update is applied to the pheromone on the edges belonging to the best-so-far solution using the following equation:

$$\tau_{ij}^{new} = (1 - \rho)\tau_{ij}^{current} + \rho(\Delta\tau_{ij}^{bs}) \quad \forall (i, j) \in T^{bs}, \tag{4}$$

where $\Delta\tau_{ij}^{bs}$ is additional pheromone, p is the global evaporation rate and T^{bs} is the best-so-far path. The additional pheromone is defined by

$$\Delta\tau_{ij}^{bs} = \begin{cases} \frac{1}{L_{bs}} & \text{if arc is in the path of } T^{bs}, \\ 0 & \text{otherwise}, \end{cases} \tag{5}$$

where L_{bs} is the total length of the best-so-far solution. The ACS algorithm usually terminates after a specific number of iterations or when the best-so-far solution achieves a desired value. The general steps of the ACS algorithm are summarized in Table I.

III. Opposition-Based ACS

One important idea of OBL is that it can be used to effectively explore different regions in the solution space to improve accuracy and convergence rates [16], [17]. The general idea can be applied in many different ways. In Differential Evolution algorithms, a mathematically opposite population is generated and combined with the original population and

TABLE I
ACS ALGORITHM

```
Initialize pheromone matrix, $\tau$, values to $\tau_o$
Repeat until termination condition is satisfied
  Repeat until solution is constructed (for each ant $k$):
    Pick next city $j$
    Apply local pheromone update
  Update best-so-far if necessary and apply global update
```

only the best individuals are kept for further optimization [10]. In this approach, opposition is a way to reach far points in the solution space, which may have a greater fitness. In Neural Networks, opposition has been successfully applied by incorporating opposite transfer functions [18]. This implementation leads to faster convergence. Finally, in reinforcement learning, opposition is used to accelerate the learning process by additional update for opposite states and opposite actions [16], [14].

In the case of ACS, it was determined that opposition can be applied in the construction phase or the update phase of the ACS algorithm. The concept of opposition as presented in [16], [17] serves as a starting point for the extensions proposed in this paper. The ACS algorithm does not work with complete solutions, so it is very complicated to establish an opposite solution. Thus, the idea was to think of opposition as a way of increasing the coverage of the solution space.

The ACS can be modified in the construction phase and the update phase. This paper proposes extensions to the construction phase of the ACS. The construction phase can be modified in two ways:

1) Change the decision;
2) Change the parameters of the decision (pheromone or heuristic).

The first three proposed extensions follow the first type of modification and are based on the idea of paired ants searching the space. By pairing ants and synchronizing their construction, one can reduce the randomness to achieve more accuracy. The other two methods follow the second possible modification and work with opposite pheromone values. While the two last approaches resemble the *explorer ants* proposed in [9], they differ in that the entire colony is subject to the possibility of using opposite pheromone content. Also, the opposite pheromone is not always activated. In the following subsections, these five versions will be described in detail.

A. Synchronous Opposition

The synchronous opposition approach is the most rigid in terms of synchronicity. The ants of the colony are paired and each pair follows a similar construction behaviour. The first ant (*leading-ant*) selects its next city as usual, but the second ant (*opposite-ant*) picks its next city based on the selection of the *leading-ant*. If the *opposite-ant* was on the same city as the *leading-ant*, it selects the opposite city. The opposite city is determined by calculating the rank of the city selected by the *leading-ant* and assigning the city with the opposite rank to the *opposite-ant*. The cities are ranked based on the combination of pheromone content and heuristic on the edge connecting them to the current city.

In contrast, if the *opposite-ant* was located on a different city, then it will mimic the decision make by the *leading-ant*. In other words, the *opposite-ant* will select a city with the same rank as the one selected by the *leading-ant*. It is important to note that the same rank does not necessarily mean the same city, because as the construction progresses, the *leading-ant* and the *opposite-ant* will have visited different cities. This procedure is followed for the entire construction phase. Each pair of ants starts on a randomly selected city.

Through opposition, the *opp-ants* diverge from their corresponding *leading-ant*, which helps guide the ants into different areas of the solution space, for a better coverage of the solution space. Additionally, it maintains a constant synchronous relationship between the two ants which reduces the randomness of the selection process. The Synchronous Opposition algorithm is included in Table II.

B. Free Opposition

The free opposition approach retains the opposite selection element from the previous synchronous method, but relaxes its synchronicity aspect. Thus, as with the Synchronous Opposition extension, when the *opposite-ant* is located on the same city as the *leading-ant*, the *opposite-ant* will move to the opposite-ranking city. However, in the other case, the *opposite-ant* will behave exactly like the *leading-ant* by selecting the next city using the pseudorandom rule (see Eq. (1)). This method was implemented to examine the effect of removing the rigid synchronicity between the two ants. Nevertheless, since ACS has a greedy selection process, the *leading-ant* and the *opposite-ant* will both often select the highest ranking city. Table II includes the Free Opposition extension of the ACS algorithm.

C. Free Quasi-Opposition

The third synchronous algorithm is similar to the Free Opposition extension. When the *leading-ant* and the *opposite-ant* are located on different cities, they will both use the pseudorandom rule (see Eq. (1)) to select their next city. In the case when the two ants are on the same city, the *opposite-ant* will also use the pseudorandom rule, but it will not be allowed to select the city that the *leading-ant* chose. This extension tries to increase the exploration of the solution space by restricting some of the choices by the *opp-ants* and guiding them into different directions. However, in contrast to the Synchronous Opposition and Free Opposition methods, the *opposite-ant* is still able to highly ranked edges when it is on the same city as the *leading-ant*. Table II includes the Free Quasi-Opposition extension of the ACS algorithm.

D. Opposite Pheromone per Node (OPN)

The OPN extension to ACS affects the pheromone value used by the ants to make their selection. Every time an

TABLE II

PAIR-BASED ALGORITHMS (SYNCHRONOUS OPPOSITION, FREE OPPOSITION, FREE QUASI-OPPOSITION)

```
Initialize $\tau$ matrix values to $\tau_o$
Repeat until termination condition is satisfied
  Repeat until solution is constructed (for each ant $k$)
    IF ant $k$ is leading-ant
      Pick next city $j$
    ELSE
      IF opposite-ant was on SAME city as leading-ant
        Synchronous Opposition: Pick opposite-rank city
        Free Oppostion: Pick opposite-rank city
        Free Quasi-Opposition: Use pseudorandom rule (but cannot pick same city as leading-ant)

      ELSE
        Synchronous Opposition: Pick same-rank city
        Free Oppostion: Pick next city $j$
        Free Quasi-Opposition: Pick next city $j$
      Apply local pheromone update (see 3)
  Update best-so-far if necessary and apply global update
```

TABLE III

OPPOSITE PHEROMONE PER NODE ALGORITHM

```
Initialize $\tau$ matrix values to $\tau_o$
Repeat until termination condition is satisfied
  Repeat until solution is constructed (for each ant $k$):
    IF $\breve{\lambda} < \breve{\lambda}_o$
      Calculate opposite pheromone values, $\breve{\tau} = \tau_o + \frac{1}{L_{bs}} - \tau$
      Pick next city $j$
    ELSE
      Pick next city $j$ (regular selection rule)
    Apply local pheromone update
  Update best-so-far and apply global update
```

TABLE IV

OPPOSITE PHEROMONE ON EDGE ALGORITHM

```
Initialize $\tau$ matrix values to $\tau_o$
Repeat until termination condition is satisfied
  Repeat until solution is constructed (for each ant $k$):
    FOR each available city $j$
      IF $\breve{\lambda} < \breve{\lambda}_o$
      Use opposite pheromone for edge $ij$, $\tau_{ij} = \tau_o + \frac{1}{L_{bs}} - \tau_{ij}$
    END for
      Pick next city $j$ (using new $\tau_{ij}$)
    Apply local pheromone update (see Eq. (3))
  Update best-so-far and apply global update
```

ant k has to select a city from the available cities, the pheromone content used for their decision will depend on an *opposite rate*, $\breve{\lambda}_o$. Given $\breve{\lambda}$ is a uniform random number, if $\breve{\lambda} < \breve{\lambda}_o$, then the ants select their next city using the opposite pheromone content, $\breve{\tau}$,

$$j = \begin{cases} \underset{l \in \mathrm{N}_i^k}{\operatorname{argmax}} \left\{ \breve{\tau}_{il} [\eta_{il}]^\beta \right\} & \text{if } q < q_o, \\ J & \text{otherwise,} \end{cases} \tag{6}$$

$$p_{ij}^k = \frac{[\breve{\tau}_{ij}][\eta_{ij}]^\beta}{\sum_{l \in \mathrm{N}_i^k} [\breve{\tau}_{il}][\eta_{il}]^\beta} \text{if } j \in N_i^k \tag{7}$$

where $\breve{\tau}$ is defined by

$$\breve{\tau} = \tau_o + \frac{1}{L_{bs}} - \tau, \tag{8}$$

where τ_o represents the initial pheromone deposit and L_{bs} is the length of the best-so-far path. These values are used to determine the opposite pheromone content because they bound the possible pheromone deposit. Given the governing equations of ACS, the pheromone content is bounded by the initial pheromone deposit and the global optimal value [6]. Furthermore, the pheromone of all the available edges will be modified. In the other case, when $\breve{\lambda} > \breve{\lambda}_o$, the ant will select the next city using the original pheromone content. The local update and global updates are were not altered. This pheromone-centred extension differs from the *explorer ants* method proposed by Montgomery and Randall [9]. In their approach, only a small portion of the colony used the anti-pheromone. Additionally, these *explorer ants* always used the anti-pheromone in their selection. In the method proposed in this paper, all ants have the opportunity to use the opposite and the affected decisions vary from ant to ant and from iteration to iteration. Table III describes the OPN extension on the ACS.

E. Opposite Pheromone per Edge (OPE)

The second pheromone extension, OPE, is a modification of the OPN method. The ants also have the possibility to use the opposite pheromone value to make their decision. However, the opposite rate, $\breve{\lambda}_o$, is applied to each individual edge of the decision instead of applying it to all the edges connected to the current city. Table IV describes the OPE extension on the ACS. The ants use the pseudorandom selection rule (see Eq. (1)) to make their decision. The pheromone of each edge is determined by

$$\tau_{ij} = \begin{cases} \tau_{ij} = \tau_o + \frac{1}{L_{bs}} - \tau_{ij} & \text{if} \breve{\lambda} < \breve{\lambda}_o, \\ \tau_{ij} & \text{otherwise.} \end{cases} \quad (9)$$

IV. Experimental Results

A. Experimental setup

The five extended algorithms were compared to the ACS algorithm on 4 different TSP instances, namely eil51, eil76, kroA100 and d198 [12]. Table V provides more details about each problem. They are all symmetric TSP instances of geographical nature. The two opposite-action algorithms were implemented in MATLAB, while the other three alternatives and the regular ACS were coded in the C language based on the code developed by T. Stützle [15]. The algorithms solved the instances using real-valued distances, the optimal values thus differ from the integer-based optimal tours.

The algorithms terminated after 5000 iterations. The parameters of all the algorithms were set to the same values, $\beta = 2$, $p = 0.1$, $\xi = 0.1$, $m = 10$, $q_o = 0.9$. These values were selected based on other research done involving ACS and TSP [5], [9]. Each algorithm completed 100 trials for the three smaller instances and 70 trials for the 198-city problem. The *opposite rate*, $\breve{\lambda}_o$, for the OPN and OPE algorithms was set to 0.01 and 0.001 respectively.

B. Results

The performance of each algorithm was evaluated in terms of the quality of the solution and the iteration when the best solution was found. The Wilcoxon rank sum (or Mann-Whitney) test was used to compare the results [7]. If the result of the test comparing two samples is significant ($p < 0.05$) we can accept the alternative hypothesis that there is a difference between the median of the two samples. A multiple comparison adjustment was not included because the comparisons were only done between normal ACS and the particular modified version. Table VI summarizes the accuracy results for the different algorithms. The median, minimum, maximum and number of times the optimal value was achieved are reported. Table VII includes results on the iteration number when the final solution of the algorithm was found.

The Synchronous Opposition and Free Opposition algorithms found significantly worse solutions than ACS. However, the Free Opposition approach was still able to find the optimal solutions for the three smaller TSP instances. When comparing the number of iterations to achieve their best-so-far solution, Synchronous Opposition took significantly more iterations for the 100-city problem and significantly less iterations for the 198-city problem ($p < 0.01$). The Free Opposition method had significantly more iterations in both the 76-city ($p < 0.05$) and 100-city instances ($p < 0.01$). These results seem to indicate that the two methods have a lower convergence rate, as they are unable to find the optimal solution for the smaller city instances even with a larger number of iterations. When considering the 198-city problem, the smaller number of iterations is an indication that the Synchronous Opposition algorithm has difficulty improving and remains in a local optima.

The Free Quasi-Opposition extension to ACS was more successful than the other two pair-based approaches. Its performance is equivalent to the normal ACS with no significant differences in their solution quality. There were also no significant differences in the number of iterations required to achieve the final solution. These results suggest that this extension did not have enough impact on the ACS algorithm. However, the Free Quasi-Opposition algorithm was able to find optimal solutions for the three smaller instances.

The two pheromone-based approaches had better solutions than the other three extensions. The OPN alternative had comparable results with the normal ACS method, but it provided better solutions ($p < 0.05$) for the 76-city case. When looking at the performance of all the algorithms with respect to the number of times the optimal solution was achieved, it seems that the 76-city case is a more complex problem than the 100-city one. Thus, the improvement achieved by the OPN extension is that much more important. The OPN achieved the optimal solutions more frequently than the normal ACS for the 76- and 100-city problems. When looking at the iteration measure, OPN had no major differences with ACS. Overall, the results indicate that forcing the ants to use opposite pheromone levels for some of their decisions can lead to better solutions. Further investigation of the effects of the rate at which the ants use the opposite-pheromones is included in section IV-C.

The OPE approach had no significant difference in solution quality compared to the ACS algorithm for all the instances. This may suggest that the extension is having a very small impact on the path construction phase. However, OPE was still able to find the optimal solution twice in the 51-city instance. The results of the number of iterations to reach their

TABLE V
OVERVIEW OF TSP INSTANCES

Instance	#Cities	Optimal Tour (real-valued)
eil51	51	428.87
eil76	76	544.37
kroA100	100	21285.44
d198	198	15808.65

TABLE VI
RESULTS FOR GLOBAL VALUE ACHIEVED COMPARING ACS AND OPPOSITION-BASED EXTENSIONS

Instance	Measure	ACS	SyncOpp	FreeOpp	FreeQOpp	OPN	OPE
eil51	Median	429.48	435.34	433.70	430.24	429.48	429.89
	Min	428.87	428.87	428.87	428.87	428.87	428.87
	Max	437.80	449.65	446.54	445.81	439.61	442.79
	#Opt	4	0	1	5	3	1
eil76	Median	553.77	561.53	561.38	553.83	552.37	554.82
	Min	545.97	548.70	544.37	545.39	544.37	546.24
	Max	565.16	576.94	573.99	563.95	562.38	566.95
	#Opt	0	0	1	0	2	0
kroA100	Median	21456.98	21755.14	21708.75	21472.67	21445.23	21399.49
	Min	21285.44	21316.38	21285.44	21285.44	21285.44	21285.44
	Max	22499.74	22054.19	22685.28	22584.25	22460.97	22145.66
	#Opt	7	0	2	7	8	9
d198	Median	16143.66	16781.00	16712.46	16127.33	16150.21	16579.46
	Min	15919.55	16328.62	16266.11	15955.87	15947.49	16042.87
	Max	16936.53	179433.3	17523.20	17062.47	16864.54	17671.37
	#Opt	0	0	0	0	0	0

TABLE VII
MEDIAN FOR FINAL ITERATION COMPARING ACS AND OPPOSITION-BASED EXTENSIONS

Instance	ACS	SyncOpp	FreeOpp	FreeQOpp	OPN	OPE
eil51	2447.5	1901	2910	2181	2420	2774.5
eil76	3340	2986	3660.5	3582	3047	3947.5
kroA100	1939.5	3003	3815.5	2312	2154	2729.5
d198	4480	4343	4718	4568.5	4621	4778

final solution revealed that OPE required significantly more iterations than ACS for the 76- and 100-city problems ($p < 0.05$). The results indicate that OPE has a lower convergence rate than normal ACS. Further work is required to determine the effects of varying the *opposition-rate* and the way the rate is applied during the optimization.

C. Varying opposition-rate of OPN

Given that the OPN algorithm had a better overall performance than all the other proposed extensions, more experiments were conducted with three other versions of the algorithm. In each version, the *opposite rate* ($\breve{\lambda}_o$) was changed to 0.05, 0.1 and 0.3. This was an attempt to study the effect of increasing the number of times ants would use the opposite pheromone value. Table VIII summarizes the solution quality results for the normal ACS algorithm, the original OPN and the three new versions.

When $\breve{\lambda}_o = 0.05$, the accuracy results are the best. The algorithm performed better than the normal ACS and the original OPN for the three smaller instances and the median difference was statistically significant ($p < 0.05$). Additionally, it was able to reach the optimal value more frequently. In the case of the 100-city instance, the algorithm reached the optimal value 13 times compared to 7 and 8 times for the ACS and original OPN respectively. For $\breve{\lambda}_o = 0.1$, the accuracy results were only slightly better than the original OPN version and the accuracy for the 76-city instance was better than for ACS ($p < 0.05$) with statistical significance. In contrast, the accuracy for OPN ($\breve{\lambda}_o = 0.1$) on the 198-city problem was significantly worse than ACS and the original OPN. Finally, setting $\breve{\lambda}_o = 0.3$ resulted in worse results for all test instances.

The results indicate that the use of opposite pheromone content for some decisions can improve the accuracy of the solutions. It is noted that a very low value for the opposition-rate will improve the results but not significantly. The results did improve when the opposition-rate was increased from 0.01 to 0.05 and 0.1. However, the performance was not as good when $\breve{\lambda}_o = 0.1$, suggesting that increasing the opposition-rate too much results in a drop in accuracy. This is supported by the fact that the performance was lower when the opposition-rate was set to 0.3. Overall, there is an indication that $\breve{\lambda}_o = 0.05$ is a reasonable value. Nevertheless, it would be important to further investigate the effects of

TABLE VIII

RESULTS FOR GLOBAL VALUE OF OPN WITH VARYING OPPOSITION-RATE ($\breve{\lambda}_o$)

Instance	Measure	ACS	OPN(0.01)	OPN(0.05)	OPN(0.1)	OPN(0.3)
eil51	Median	429.48	429.48	429.05	429.30	431.48
	Min	428.87	428.87	428.87	428.87	428.87
	Max	437.80	439.61	437.50	435.70	437.92
	#Opt	4	3	5	4	2
eil76	Median	553.77	552.37	551.07	551.24	558.68
	Min	545	544.37	544.37	544.37	549.65
	Max	565.16	562.38	558.65	564.06	576.9
	#Opt	0	2	3	2	0
kroA100	Median	21456.98	21445.23	21386.47	21392.81	21982.30
	Min	21285.44	21285.44	21285.44	21285.44	21539.87
	Max	22499.74	22460.97	22826.44	22381.91	23223.99
	#Opt	7	8	13	9	0
d198	Median	16143.66	16150.21	16267.34	16734.15	17425.06
	Min	15919.55	15947.49	16061.74	16287.07	16890.30
	Max	16936.53	16864.54	16716.67	17117.37	17768.70
	#Opt	0	0	0	0	0

varying the opposition-rate during the optimization. In fact, this may lead to an improvement in performance for larger instances.

V. CONCLUSIONS

This study introduced five extensions to the ACS algorithm. The extensions incorporated opposition-based concepts as an attempt to improve the quality of solutions and convergence rate of ACS. It was an opportunity to explore the applicability of opposition ideas to ACO.

Three extensions involved pairing the ants (*leading-ant* and *opposite-ant*) and synchronizing their construction:

- In the first, Synchronous Opposition, if the *opposite-ant* is on the same city as the *leading-ant*, it selects a city with opposite-rank; otherwise it selects a city with the same rank.
- The second approach, Free Opposition, only differs from the first when the ants are on different cities. In that case, the *opposite-ant* will select cities using the normal selection rule.
- The third pair-based extension, Free Quasi-Opposition, is similar to the Free Opposition method, except when the paired ants are on the same city. In that situation, the *opposite-ant* selects the next city using the regular rule, but it is not allowed to select the same city as the *leading-ant*.

The other two extensions to ACS involved the use of opposite-pheromone.

- In one approach, Opposite Pheromone per Node (OPN), the ants would, at a specified frequency, make the selection of their next city by using an opposite-pheromone instead of the regular pheromone value for all the available cities.
- The other approach, Opposite Pheromone per Edge (OPE), is the same as OPN, except that the rate at which the opposite-pheromone is used is applied to each individual available city instead of the decision as a whole.

The regular ACS and proposed extensions were tested on four TSP instances. The experimental results showed that

1) with the appropriate *opposition-rate*, the OPN method generates better solutions than regular ACS for the three smaller instances. The improvements were statistically significant.
2) the Synchronous Opposition and Free Opposition approaches performed worse than all the other extensions and the normal ACS. The Free Quasi-Opposition and OPE methods had comparable performance with ACS, which may be an indication that the modifications can potentially lead to an improvement.

Consequently, it would be interesting to fully investigate the extent of the exploration done by each algorithm. While the OPN extension proved successful for the three small TSP instances, more work is required to fully explore the benefits of this extension. The results show that the pheromone content is a key element in the solution creation. Thus, using more complex pheromone behaviour can lead to a better coverage of the search space. There is a possibility that varying the *opposition-rate* during the optimization might positively affect the results. Additionally, it would be important to explore the concept of opposition in combination with local search, since the addition of local search greatly improves the performance of algorithms. Computational expense differences should also be evaluated. Future work will also involve the investigation of new ways of using the pheromone deposits.

REFERENCES

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems.* New York: Oxford University Press, 1999.

[2] O. Cordón, I. F. de Viana, F. Herrera, and L. Moreno, “A New ACO Model Integrating Evolutionary Computation Concepts: The Best-Worst Ant System,” in *Proc. of ANTS'2000 - From Ant Colonies to Artificial Ants: Second Interantional Workshop on Ant Algorithms,* Brussels, Belgium, pp. 22-29, 2000.

[3] M. Dorigo, *Optimization, Learning and Natural Algorithms (in Italian)*, PhD Thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.

[4] M. Dorigo, V. Maniezzo, and A. Colorni, “The Ant System: Optimization by a Colony of Cooperating Agents,” *IEEE Trans. Systems*, *Man*, and *Cybernetics*, vol. 26, pp. 29-41, 1996.

[5] M. Dorigo, and L. M. Gambardella, “Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem,” *IEEE Transactions On Evolutionary Computation,* vol. 1, no. 1, pp. 53-66, 1997.

[6] M. Dorigo and T. Stützle, *Ant Colony Optimization.* Cambridge, Massachusetts: The MIT Press, 2004.

[7] M. Hollander, and D.A. Wolfe, *Nonparametric Statistical Methods.* Wiley, 1973.

[8] S. Iredi, D. Merkle, and M. Midderndorf, “Bi-Criterion Optimization with Multi Colony Ant Algorithms,” in *P59roc. First Int. COnf. on Evolutionary Multi-Criterion Optimization (EMO’01),* LNCS 1993, pp. 359-372, 2001.

[9] J. Montgomery and M. Randall, “Anti-Pheromone as a Tool for Better Exploration of Search Spaces,” in *Proc. 3rd Int. Workshop on Ant Algorithms,* ANTS2002, Brussels, Belgium, September 2002, pp. 100-110.

[10] S.Rahnamayan, H.R.Tizhoosh, and M.M. Salama, “Opposition-Based Differential Evolution Algorithms,” in *Proc. IEEE Congress on Evolutionary Computation,* Vancouver, July 16-21, 2006, pp. 7363-7370.

[11] M. Randall and J. Montgomery, “The Accumulated Experience Ant Colony for the Travelling Salesman Problem,” in *Proceedings of Inaugural Workshop on Artificial Life,* Adelaide, Australia, pp. 79-87, 2001.

[12] G. Reinelt, “TSPLIB - A traveling salesman problem library,” *ORSA J. Comput.,* vol. 3, pp. 376-384, 1991.

[13] R. Schoonderwoerd, O.E. Holland, J.L. Bruten, and L.J.M. Rothkrantz, “Ant-Based Load Balancing in Telecommunications Networks,”, *Adaptive Behavior,* vol. 2, 1996, pp. 169-207.

[14] M. Shokri, H.R. Tizhoosh and Mohamed Kamel, “Opposition-Based Qλ) Algorithm,” in *Proc. IEEE International Joint Conf. on Neural Networks (IJCNN),* Vancouver, July 16-21, 2006, pp. 646-653.

[15] T. Stützle, Ant Colony Optimization, *Public Software,* June 14, 2004. http://iridia.ulb.ac.be/ mdorigo/ACO/aco-code/public-software.html

[16] H.R. Tizhoosh, “Opposition-Based Learning: A New Scheme for Machine Intelligence”, in *Proc. Int. Conf. on Computational Intelligence for Modelling Control and Automation - CIMCA’2005,* Vienna, Austria, vol. I, pp. 695-701, 2005.

[17] H.R. Tizhoosh, “Opposition-Based Reinforcement Learning,”, *Journal of Advanced Computational Intelligence and Intelligence Informatics,*, vol. 10, no. 4, pp. 578-585, 2006.

[18] M. Ventresca and H.R. Tizhoosh, “Improving the Convergence of Backpropagation by Opposite Transfer Functions,” in *Proc. IEEE International Joint Conf. on Neural Networks (IJCNN),* Vancouver, July 16-21, 2006, pp. 9527-9534. hebibliography