Reactive Memory Model for Ant Colony Optimization and Its Application to TSP

Rafid Sagban[1]; Ku Ruhana Ku Mahamud[2]; Muhamad Shahbani Abu Bakar[2]
[1] Computer Science Dept., University of Babylon,
Babylon, Iraq
rsagban@uobabylon.edu.iq
[2] School of Computing, College of Arts and Sciences, Universiti Utara Malaysia,
Sintok, Kedah, Malaysia
{ruhana, shahbani}@uum.edu.my

***Abstract*—Ant colony optimization is one of the most successful examples of swarm intelligent systems. The exploration and exploitation are the main mechanisms in controlling search within the ACO. Reactive search is a framework for automating the exploration and exploitation in stochastic algorithms. Restarting the search with the aid of memorizing the search history is the soul of reaction. It is to increase the exploration only when needed. This paper proposes a reactive memory model to overcome the limitation of the random exploration after restart because of losing the previous history of search. The proposed model is utilized to record the previous search regions to be used as reference for ants after restart. The performances of six (6) ant colony optimization variants were evaluated to select the base for the proposed model. Based on the results, Max-Min Ant System has been chosen as the base for the modification. The modified algorithm called RMMAS, was applied to TSPLIB95 data and results showed that RMMAS outperformed the standard MMAS.**

***Index Terms*— Ant colony optimization, reactive search, exploration mechanism, exploitation mechanism.**

I. INTRODUCTION

Ant colony optimization (ACO) algorithms are multi-agent systems utilized for solving hard combinatorial optimization problems. Despite being one of the youngest meta-heuristics, there is a large number of applications of ACO algorithms, such as: engineering design, topology optimization and structural optimization in electronics, aerodynamics, fluid dynamics, telecommunications, automotive, and robotics; machine learning and data mining in bioinformatics and finance; system modeling, simulation and identification in chemistry, physics, and biology; control, signal, and image processing; planning in routing problems, robot planning, scheduling and production problems, logistics and transportation; and supply chain management [1, 2].

In ACO algorithms, the behavior of each agent is inspired from the food foraging behavior of ants. The way that ant workers utilize to find their food sources was transmitted into a probabilistic model. It soon stimulated the inventing of the first ACO algorithm called Ant System (AS). Based on the simple schema of ant system algorithm, a number of other AS variants were developed [3]. The core aspect of this continuous improvement is to enhance its searching process. It is all about achieving a proper balance between exploration and exploitation (E&E). Designing a well-balanced ACO algorithm is needed to find high quality solutions for the problem [4]. However, existing exploitation procedures lead the search to be stagnated while the exploration procedures lead to slow convergence. MMAS algorithm, as one of the best of AS variants, proposed several E&E mechanisms as exemplified by elitisms, max-min bounds, smoothing, trail learning or restarting. The basic idea of all said mechanisms is modifying a probabilistic memory model called pheromone memory. This memory model governs the way that ants traverse the search space and summarize their searching experiences.

Reactive search framework [5] is an emerging research area for improving the internal behavior of any meta-heuristic. The word "react" hints to the ready response to events during the search. This internal flexibility is maintained by harnessing the past history of the search for automating E&E balance. The critical aspects for the reactive search are the memory model and the restart mechanism. The former aspect is to support the exploitation concept by memorizing the good search regions. The later one is to support the exploration by finding new regions, as needed. In this paper, the two aspects have been tested to characterize how much they can be fitted in the current E&E mechanisms. The experiments showed that utilizing the two mechanisms without clear methodology may worsen the performance of ACO algorithm. Apart from that, the analyzing results showed that MMAS is a good candidate to be integrated with the proposed reactive procedures and is more convenient to inherit most of the reactive search attributes. However, when emerged with restarts, it suffers a random exploration with no guarantee that the visited regions will not be traversed again and again. This side effect will produce an over-exploration state. Moreover, after restart, the pheromone trail will lose the accumulated information (i.e. population experience) about the previous search [6]. Hence, the main questions associated with these problems are: i) how the ACO variants behave against reactive procedures? ii) what are the requirements for a reactive model to record the history of the search? iii) how to integrate the

proposed model with reactive procedures in the best ACO variant?

The structure of the paper is organized as follows: Section II reviews the existing memory models while Section III describes the proposed model. The evaluation of the proposed model is presented in Section IV and the conclusion and future work are given in Section V.

II. MEMORY MODELS IN ACO

The artificial ants in ACO algorithms are optimization agents used to mimic the ants' behavior in nature. However, these agents have several major differences with the real ants which have some memories known as the pheromone model, to record the experience of the colony during the search. A colony of ants start constructing their paths step by step until they find solutions to the problem under hand. The solutions will be evaluated to pick the optimized solution. Once a new solution is found, the objective function will be minimized or maximized accordingly. Subsequently, the memory of the colony can be changed either online, offline or by resetting the pheromone values. This way of pheromone management influences the diversification (i.e. exploration) and intensification (i.e. exploitation) of the search process. This category of mechanisms suffers a stagnation problem because the algorithm run time is consumed with no ability to improve the optimality of the current solution.

Most of the exploration and exploitation mechanisms in classic ACO algorithms fall under the pheromone management class as exemplified in [7, 8, 9, 6]. This is not confined to the old algorithms but extends to several enhanced ACO algorithms [10, 11, 12]. The second class of mechanisms focused on the interaction among pheromone memories instead of using only one distributed memory [13, 14]. The third class added a new kind of memory called the population memory. This contributed to a very fast convergence. New research directions try to analyze this mechanism in order to understand its behavior [15]. In Angus [16], a new population based technique for exploration called niching. The fourth class puts the ACO in relation with other exploration and exploitation mechanisms applied in other approximation methods. This can be justified by the noteworthy shift towards the hybridization of meta-heuristics with other techniques for optimization [17]. The focus of research has changed from being rather algorithm-oriented to being more problem-oriented. The focus is on solving the problem at hand in the best way possible, rather than promoting a certain meta-heuristic. The fifth class is the parameterization. It is strongly related to the exploration and exploitation mechanism. Parameters are the components that allow the users of any algorithm to adjust its exploration/exploitation manually. Alternatively, Eiben and Smit [18] proposed a general framework for tuning parameters automatically based on three factors: the problem, the algorithm and the tuner. Furthermore, the role of the tuner can be excluded using intelligent parameter controllers. Stützle et al. [19] reviewed the existing parameter controllers in ACO literature. In general, they can be classified, based on their feedback mechanisms, pre-scheduled and adapted approaches. The last class known as reaction is based on the idea that the search process governed by machine learning mechanisms. In Khichane et al. [20] the exploration and exploitation mechanism suggested can be automated in some way based on the history of search in self-adaptive fashion.

In the aforesaid exploration and exploitation mechanisms, the adaptive memory is the substrate. The memory in ACO algorithm is represented by the pheromone model. It is a parameterized probability distribution over the solution space. Using the model, the ants construct solutions as in equation (1) below:

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}.\mu_{ij}^{\beta}}{\sum_{c_{il}\in N(S^P)} \tau_{il}^{\alpha}.\mu_{il}^{\beta}} & if\ c_{il} \in N(S^P) \\ 0 \quad otherwise \end{cases}, \tag{1}$$

where τ_{ij} is the pheromone value adjusted by the parameter α and μ_{ij} is the heuristic value, which is given by: 1/distance (i, j). μ is adjusted by the parameter β. The specification of N (s^p) depends on the solution construction mechanism. The constructed solutions modify the pheromone values as in equation (2) given by:

$$\tau_{ij} = (1-\rho).\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \tag{2}$$

where the $\Delta\tau_{ij}^{k}$ is determined by:

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & if\ ant\ (k)\ used\ edge\ (i,j) in\ its\ tour, \\ 0 & otherwise, \end{cases} \tag{3}$$

Q is a constant and L_k is the length of the tour constructed by ant (k) while parameter $\rho \in [0, 1]$ is a pheromone trail decay coefficient (i.e. evaporation rate). Once the ants finished pheromone updating, they will die (i.e. current iteration has been finished). In TSP (travelling salesman problem), the solution construction mechanism restricts the set of traversable edges (i.e. N (s^p)) to the set of untraversed edges by k^{th} ant.

The way that the two processes (i.e. constructing solutions and modifying memory) interact determines the explorative and exploitative behavior of the algorithm. For example, allowing elite ants to modify memory leads the history search to be over-exploited. In contrary, allowing bad performing ants to modify memory leads the search space to be over-explored. The later choice may help in finding a good solution but it causes a slow convergence in its final outcome. To achieve some trade-off, this study harnesses the reactive search [5] as a generic framework to manage the exploration versus exploitation dilemma in ACO.

III. THE PROPOSED REACTIVE MODEL

This section will introduce two reactive components. They are reactive restart and reactive memory. Figure 1 showed the steps toward a reactive ACO model.

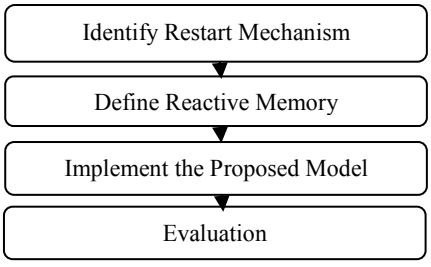

Fig.1. The proposed reactive approach.

In the proposed approach, the reaction of memory utilized for exploitation and the reaction of restart will increase the exploration as needed. The pheromone values are modified in an online way biasing the decisions of ants in the next iterations. In this way, the ants' decisions will not be similar and the optimization process will improve. When all ants are converged early, the restart mechanism will force them to search other space regions.

In identifying the restart mechanism, let us consider an ACO algorithm suffers an early convergence to some solution at time *t*. The algorithm needs to restart the search in the hope of escaping this situation in order to improve the quality of the solution. This study also aims to identify a suitable restart mechanism. Two feedback mechanisms used to indicate the optimal restarting point are the acceptance criteria and branching factor. The acceptance criteria are calculated using equation (4) given as:

$$Restart(S_{gb}, S_k, history) = \begin{cases} -rs \ if \ f(S_k) < f(S_{gb}) \\ +rs \ if \ f(S_k) \geq f(S_{gb}) \ and \\ i - i_{last} > \epsilon \end{cases} \quad (4)$$

where *+rs* indicates that the convergence happened when the new solutions S_k since last best restart i_{last} is not optimized for last ϵ iterations (e.g. 250 iterations). The average branching factor counts the number of factors greater than $\tau_{min} + \gamma\ (\tau_{max} - \tau_{min})$ in the current node in the construction graph, then counts the average of all the counted factors.

The reactive memory can be defined by having the following assumptions. For some arcs, the trail intensity is decreased because of the evaporation influence. These arcs are recorded during the optimization process. The history of search can be recorded in an exploration and exploitation component called reactive memory denoted by LB (lower bound). This will enable the definition of the evaporation rule that uses the new component as follows:

$$Evaporate\ (LB, T, \tau_{min}) = \begin{cases} LB \leftarrow lb_0 & if \tau_{ij} < \tau_{min} \\ LB \leftarrow lb_1 & if \tau_{ij} \geq \tau_{min} \end{cases} \quad (5)$$

The standard evaporation rule in MMAS is given by

$$\tau_{ij} = (1 - \rho).\tau_{ij} \quad \forall\ \tau_{ij} \in T \quad (6)$$

In comparing the two equations, the new rule harnesses LB matrix as a history of search component. Thus, the new model is helpful for increasing the quality of solution obtained by MMAS.

In implementing the proposed model, the MMAS has been chosen to be used as the basis for the proposed ACO variant. The proposed model is called *RMMAS*. In this model, the reactive memory is initialized to LB $\stackrel{def}{=}[lb_0]$. The new memory will not interrupt the optimization progress unless new indications are provided by the feedback mechanism. Once the search stagnates, the quality of later generated solutions will not improve. Acceptance criteria and lambda branching factor in some way measure the convergence of the algorithm in each of the 100 iterations. Meanwhile, the reactive memory will record all the arcs below the lower bound of pheromone trails. In this way, unexplored regions in the current search are shifted to the next search. The history of search is indicated by a new heuristic lb_{ij}. The ability of ants to remember their previous search will influence their future decisions according to the following decision rule.

$$p_{ij}^k = \begin{cases} \dfrac{\tau_{ij}^{\alpha}.\mu_{ij}^{\beta}.lb_{ij}}{\sum_{c_{il} \in N(S^P)} \tau_{il}^{\alpha}.\mu_{il}^{\beta}} \ if\ c_{il} \in N(S^P) \\ 0 \quad otherwise \end{cases}, \quad (7)$$

In ACO algorithm, ants utilize the pheromone trail τ_{ij}^{α} and pre-heuristics μ_{ij}^{β} to construct their tours. In this way, the availability of this information is critical for ants to decide their way. The pre-heuristics may not be available in advance for some combinatorial optimization (CO) problems such as the quadratic assignment problem. The information recorded in LB even if it is beneficial to solve problems depends on the pre-heuristic such as distances among cities in TSP. In the proposed model, ants will decide their path, will utilize three sources of information: $\tau_{ij}^{\alpha}, \mu_{ij}^{\beta}$ and lb_{ij}.When the run starts, ants will supply more information on μ_{ij} in the construction of their solution. Subsequently, new information about the search space starts to cumulate in the pheromone trails τ_{ij}. At this point of run time, ants are biased toward high pheromone intensity. Hence, the pre-heuristic information about search space is becoming ineffectual in the decision of ants. In this way, ants converge toward global optimal solutions with the risk of search stagnation. To overcome this limitation, pheromone trails are reset to τ_{max}, LB memory is reactivated and a new construction solution is utilized. These three complementary procedures represent the "react" against premature convergence that all ACO algorithms suffer from regardless of their application. Figure 2 depicts the pseudocode for the proposed model.

Algorithm: Pseudocode for RMMAS:

```
ComputeDistances ()
InitializeParameters ():
 InitializePheromoneTrails ()
InitializeLowerBounds ()
while (not terminate) do
   for k := 1 to m do
      if (no stagnation) do
        ConstructSolutions (T, C)
        else
          RestartPheromoneTrails (S_gb, S_k, History)
          ConstructSolutions (T, C, LB)
        end-else
      end-if
      S_gb ← argmin{f(S_gb), f(S_k | k:=1 to m )}
      Evaporate (LB, T, τ_min)
      DepositPheromone (T, S^gb)
   end-for
 end-while
end-procedure
```

Fig. 2. Pseducode for the proposed model

IV. EXPERIMENTAL RESULTS & DISCUSSION

To observe the performance of AS, Elitist Ant System (EAS), Rank-Based Ant System (RAS), Best-Worst Ant System (BWAS), Ant Colony System (ACS) and MMAS ACO variants with restart strategy, ten (10) experiments were conducted for each instance of the TSPLIB [21]. Running time for each experiment was set to 10 sec. This time duration is enough for ACO algorithms to produce (near) optimal solutions for problems within 50-100 cities. The experiments were conducted on Windows 8 64-bit operating system, processor Intel Core i3-3217U with CPU @ 1.80GHz, RAM 4GB.

The proposed RMMAS algorithm was coded in C language. ACO variants were selected based on ACOTSP 3.0 (Dorigo & Stützle, 2004). The parameter settings selected from the literature of each ACO variant are as follows. The number of ants (m) is equal to the number of cities except ACS where m is equal to 10. The pheromone intensity (α) and pre-heuristic distance (β) are equal to 1 and 2 respectively for all variants. Evaporation rate (ρ) is 0.5 for AS and EAS; 0.1 for RAS, BWAS and ACS; and 0.02 for MMAS. Some ACO variants have several additional parameters. The settings for these parameters are: RAS: number of ranks (r) are 6; ACS: q_0 is 0.9; local update parameter is 0.1; number of nearest neighbor cities is 20 for all ACO variants. The initial pheromone (τ_0) is set to $1/\rho * C^{nn}$ in MMAS and to $1/n * C^{nn}$ in ACS. In the original papers of AS, EAS, and RAS, it did not exactly define the value of τ_0. Hence it is set to $1/\rho * C^{nn}$.

ACO variants are tested with and without restart (+rs and –rs respectively). Those with restarts used in the experiments are as follows: i) using acceptance criteria with $\in = 250$ (refer equation 1) and initial pheromone is set to τ_0; ii) using the same setting for acceptance criteria but with initial value equal to τ_{max}; and iii) using acceptance criteria and lambda branching factor with initial value equal to τ_{max}.

Tables 1 and 2 display the results of the six (6) ACO variants on success rate and mean tests. The results showed that the performance of AS has worsened with restarts unlike elitist variants that tend to be more exploitative. With restarts, more exploration is obtained. The best performance with restarts was obtained by MMAS while BWAS was the worst without restarts. It is worth mentioning that the quality of solutions is highly influenced if restarts are used. For example, MMAS performs best with dual feedback criteria (i.e. branching factor and acceptance criteria). This behavior is due to the way of managing pheromone in the pheromone memory. The optimal solution for eil51.tsp is 426 and successful runs = number of tries terminates with optimal solution/ number of tries.

Table 1. Performance obtained for the tsp instance eil51: success rate/time test

ACO variant	Successful runs				Success time (sec)			
	-rs	*+rs*			*-rs*	*+rs*		
		τ_0	τ_{max}			τ_0	τ_{max}	
		Acceptance criteria	*Acceptance criteria*	*Branching factor + Acceptance criteria*		*Acceptance criteria*	*Acceptance criteria*	*Branching factor+ Acceptance criteria*
AS	0/10	0/10	0/10	-	-	-	-	-
EAS	0/10	2/10	0/10	-	-	3	-	-
RAS	1/10	2/10	3/10	-	0.1	0.2	0.13	-
BWAS	0/10	0/10	0/10	-	-	-	-	-
ACS	1/10	1/10	1/10	-	0.19	8.6	1.03	-
MMAS	**2/10**	**6/10**	**4/10**	**8/10**	0.67	1.7	2	**0.64**

Table 2. Performance obtained for the tsp instance eil51: best/mean test

ACO variant	Best				Mean			
	-rs	*+rs*			*-rs*	*+rs*		
		τ_0	τ_{max}			τ_0	τ_{max}	
		Acceptance criteria	*Acceptance criteria*	*Branching factor + Acceptance criteria*		*Acceptance criteria*	*Acceptance criteria*	*Branching factor+ Acceptance criteria*
AS	429	430	431	-	434	436	437	-
EAS	428	426	427	-	433	430	431	-
RAS	426	426	426	-	430	428	428	-
BWAS	450	427	429	-	468	431	435	-
ACS	426	426	426	-	428	430	427	-
MMAS	426	**426**	**426**	**426**	**427**	**427**	427	**426**

The second part of our experiments is to evaluate the performance of the proposed model. Table 3 displays the results of MMAS and RMMAS. It can be seen that the best solution and average value of RMMAS are obviously superior to MMAS. The standard deviation shows that RMMAS is

Table 3. Results of MMAS versus RMMAS

TSP instance	Optimum	MMAS			RMMAS		
		Mean	*SD*	*Best*	*Mean*	*SD*	*Best*
berlin52	7542.0	7542.0	0.00	7542.0	7542.0	0.00	7542.0
st70	675.0	677.1	1.85	675.0	**676.9**	2.88	675.0
Eil76	538.0	538.6	0.52	538.0	**538.4**	0.52	538.0
pr76	108159.0	108265.0	285.81	108159.0	**108173.9**	**47.12**	108159.0
gr96	55209.0	55671.8	74.00	55601.0	**55560.9**	**71.02**	55434.0
rat99	1211.0	1211.9	0.88	1211.0	**1211.1**	**0.32**	1211.0
KroA100	21282.0	21342.0	52.14	21282.0	**21334.4**	**47.96**	21282.0
KroB100	22141.0	22301.9	30.26	22237.0	**22294.1**	**23.29**	22237.0
KroC100	20749.0	20797.0	69.12	20749.0	**20789.1**	**68.57**	20749.0
KroE100	22068.0	22337.2	148.60	22068.0	**22268.8**	**137.21**	22068.0
rd100	7910.0	7922.5	16.21	7910.0	**7919.9**	**12.49**	7910.0

V. CONCLUSION

This paper deals with the way of using "reaction" toward automated exploration and exploitation in ACO algorithm. A definition of reactive memory model is presented, and a consequent algorithmic approach is provided. It is applied to MMAS algorithm, which is one of the best ACO algorithms where the problem of random exploration is addressed. The conclusions drawn in this sense can easily be applied for other ACO algorithms. In these first experiments, only one algorithm is considered, and no local search is applied. Both of these points will be overcome in future research. In particular, it is expected that this scheme of memorizing the history of search will contribute to the local search larger than it is in global search promoted here by the reactive restarts. The application of this scheme to other combinatorial optimization problems such as quadratic assignment problem is also an area for future work.

REFERENCES

[1] B. C. Mohan and R. Baskaran, "A survey□: Ant Colony Optimization based recent research and implementation on several engineering domain," *Expert Syst. Appl.*, vol. 39, no. 4, pp. 4618–4627, 2012.

[2] T. Stützle, M. Lopez-Ibanez, and M. Dorigo, "A concise overview of applications of ant colony," in *Wiley Encyclopedia of Operations Research andManagement Science*, J. J. Cochran, Ed. Weley & Sons Ltd., 2010.

[3] M. Dorigo, M. Birattari, and T. Stützle, "Ant colony optimization: Artificial ants as a computational intelligence technique," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, 2006.

[4] C. Blum, "ACO applied to group shop scheduling: A case study on intensification and diversification," in *Lecture notes in computer science (LNCS): Ant algorithms*, vol. 2463, M. Dorigo, G. Di Caro, and M. Sampels, Eds. Heidelberg, Germany: Springer, 2002, pp. 14–27.

[5] R. Battiti, M. Brunato, and F. Mascia, *Reactive search and intelligent optimization*. New York, USA: Springer, 2008.

[6] T. Stützle and H. H. Hoos, "MAX – MIN ant system," *Futur. Gener. Comput. Syst.*, vol. 16, no. 8, pp. 889–914, 2000.

[7] B. Bullnheimer, R. F. Hartl, and C. Straub, *A new rank based version of the ant system - A computational study*, No.1 ed. Vienna: Institute of Management Science, University of Vienna, 1997.

[8] M. Dorigo and L. M. Gambardella, "Ant colony system□: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, 1997.

[9] L. M. Gambardella and M. Dorigo, "Ant-Q: A reinforcement learning approach to the traveling salesman problem," in *The Morgan Kaufmann series in machine learning: Proceedings of the 12th international conference on machine learning*, A. Prieditis and S. J. Russell, Eds. California: Morgan Kaufmann, 1995, pp. 252–260.

[10] Y. Bin, Y. Zhongzhen, and Y. Baozhen, "An improved ant colony optimization for vehicle routing problem," *Eur. J. Oper. Res.*, vol. 196, no. 1, pp. 171–176, Jul. 2009.

[11] V. P. Eswaramurthy and A. Tamilarasi, "Hybridizing tabu search with ant colony optimization for solving job shop scheduling problems," *Int. J. Adv. Manuf. Technol.*, vol. 40, no. 9–10, pp. 1004–1015, Feb. 2009.

[12] K. R. Ku-mahamud and M. M. Alobaedy, "New heuristic function in ant colony system algorithm for optimization," in *Proceedings of MAMECTIS '13: The 15th international Conference on Mathematical Methods, Computational Techniques and Intelligent Systems*, 2013, pp. 13–18.

[13] A. Aljanaby, K. R. Ku-Mahamud, and N. M. Norwawi, "Interacted multiple ant colonies optimization framework: An experimental study of the evaluation and the exploration techniques to control the search stagnation," *Int. J. Adv. Comput. Technol.*, vol. 2, no. 1, pp. 78–85, Mar. 2010.

[14] P. Rahmani, M. Dadbakhsh, and S. Gheisari, "Improved MACO approach for grid scheduling," in *Proceedings of ICIII 2012: The international conference on industrial and intelligent information*, 2012, vol. 31, pp. 135–142.

[15] S. Oliveira, M. S. Hussin, A. Roli, and M. Dorigo, "A Detailed Analysis of the Population-Based Ant Colony Optimization Algorithm for the TSP and the QAP," no. February, 2011.

[16] D. J. Angus, "Niching ant colony optimisation," Unpublished doctoral dissertation.Swinburne University of Technology Melbourne, Australia, 2008.

[17] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli, "Hybrid metaheuristics in combinatorial optimization: A survey," *Appl. Soft Comput.*, vol. 11, pp. 4135–4151, 2011.

[18] A. Eiben and S. K. Smit, "Parameter tuning for configuring and analyzing evolutionary algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 19–31, Mar. 2011.

[19] T. Stützle, L. Manuel, P. Pellegrini, M. Maur, M. M. De Oca, M. Birattari, and M. Dorigo, "Parameter adaptation in ant colony optimization," in *Autonomous search*, Y. Hamadi, E. Monfroy, and F. Saubion, Eds. Heidelberg, Germany: Springer, 2012, pp. 191–215.

[20] M. Khichane, P. Albert, and C. Solnon, "An ACO-based reactive framework for ant colony optimization: First experiments on constraint satisfaction problems," in *Lecture notes in computer science (LNCS): Learning and intelligent optimization*, vol. 5851, T. Stützle, Ed. Heidelberg, Germany: Springer, 2009, pp. 119–133.

[21] G. Reinelt, "Benchmark-TSPLIB:A traveling salesman problem library," *ORSA Journal On Computing*, 1991. [Online]. Available: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.