Intelligent Automation & Soft Computing

ISSN: 1079-8587 (Print) 2326-005X (Online) Journal homepage: http://www.tandfonline.com/loi/tasj20

Reactive max-min ant system with recursive local search and its application to TSP and QAP

Rafid Sagban, Ku Ruhana Ku-Mahamud & Muhamad Shahbani Abu Bakar

To cite this article: Rafid Sagban, Ku Ruhana Ku-Mahamud & Muhamad Shahbani Abu Bakar (2016): Reactive max-min ant system with recursive local search and its application to TSP and QAP, Intelligent Automation & Soft Computing, DOI: 10.1080/10798587.2016.1177914

To link to this article: http://dx.doi.org/10.1080/10798587.2016.1177914

INTELLIGENT AUTOMATION & SOFT COMPUTING, 2016
http://dx.doi.org/10.1080/10798587.2016.1177914

Reactive max-min ant system with recursive local search and its application to TSP and QAP

Rafid Sagban[a], Ku Ruhana Ku-Mahamud[b] and Muhamad Shahbani Abu Bakar[b]

[a]Computer Science Dept., University of Babylon, Babylon, Iraq; [b]School of Computing, College of Arts and Sciences University, Utara Malaysia, Sintok, Kedah, Malaysia

ABSTRACT

Ant colony optimization is a successful metaheuristic for solving combinatorial optimization problems. However, the drawback of premature exploitation arises in ant colony optimization when coupled with local searches, in which the neighborhood's structures of the search space are not completely traversed. This paper proposes two algorithmic components for solving the premature exploitation, i.e. the reactive heuristics and recursive local search technique. The resulting algorithm is tested on two well-known combinatorial optimization problems arising in the artificial intelligence problems field and compared experimentally to six (6) variants of ACO with local search. Results showed that the enhanced algorithm outperforms the six ACO variants.

KEY WORDS

Optimization; combinatorial problems; metaheuristics; swarm intelligence; search algorithms; ant colony optimization; recursive local search; reactive heuristics; traveling salesman problem; quadratic assignment problem

1. Introduction

For the necessity of solving combinatorial optimization (CO) problems, metaheuristic algorithms have been invented as being either model-based or instance-based searching methods. Ant colony optimization (ACO) is a model-based metaheuristic inspired by the food foraging behavior of real ants. The CO problem is modeled into a construction graph so that stochastic search agents, called artificial ants, can perform a walk through that graph. The distinctive feature of ACO is a particular type of probabilistic model in which the graph is coupled with the agents. Iteratively, generations of ants move on graph searching feasible solutions to the CO problem under hand. Subsequently, the ants update the model in an on-the-fly reinforcement learning way, so that the search concentrates in regions containing high quality solutions. This learning scheme utilizes types of numerical values called artificial pheromone trails (Dorigo & Stützle, 2010). The way that pheromone trails are distributed, i.e. the pheromone updating functions, determines in which parts of CO search space the high quality solutions are located. In ACO, an effective searching agent can be designed if it has the ability to achieve an appropriate balance between exploiting the ants' search experience gathered so far and exploring relatively unvisited search space regions.

A large number of real-world problems in many fields can be modelled as CO problems. Despite being one of the youngest metaheuristics, ACO algorithms achieved great success in problem solving of engineering design, topology optimization, structural optimization in electronics, aerodynamics, fluid dynamics, telecommunications, automotive, robotics and artificial intelligence, signal and image processing, scheduling problems, logistics and transportation (Mohan & Baskaran, 2012; Stützle, Lopez-Ibanez, & Dorigo, 2010; Wanga, Wua, Yanga, & Liua, 2011). Moreover, this expansion in the application motivates new hybrid metaheuristics (Blum, Puchinger, Raidl, & Roli, 2011), that do not follow the traditional ACO paradigm. Recently, new hybrid methods that combines local search with ACO algorithm have received more and more attention (Gambardella, Montemanni, & Weyland, 2012). According to Battiti, Brunato, and Mascia (2008), the most successful ACO applications are the ones coupling it with local search and advanced memory features. The main goal of the hybridization is to achieve well-balanced exploration and exploitation and to exploit the complementary characteristics of manipulating neighborhood structures during search. However, the problem of exploitation in this category of local search based ACO algorithms arises when the local search procedures prematurely traverse the neighborhood structures. The search around promising regions found by previous generations of ants is not completely transferred to the future generations.

This paper tries to develop an ACO based algorithm hybrid with local search in which advanced memory features can be utilized to record information about neighborhood structures generated either by ants and local search routines. Two memory schemes are designed to transfer the said neighborhood structures over iterations; one scheme for each category. The recorded information is used to guide the search of ants in one of the most powerful variant of ACO for traveling salesman problem (TSP) and quadratic assignment problem (QAP), i.e. max-min ant system (MMAS) algorithm (Stützle & Hoos, 2000). Hence, the proposed algorithm induced MMAS. Specifically, the contributions of the paper lie in two aspects:

- Reactive heuristics are developed to be considered as local heuristics in the transition probabilistic rule of construction solution function. The arcs with low concentrated pheromone will be identified during the evaporation function at each step of pheromone update. These arcs will be recorded in the scheme of component-based

CONTACT Rafid Sagban rsagban@uobabylon.edu.iq

memory for future consideration after the search is restarted.
- A recursive local search technique is developed based on the scheme of population-based memory where previous populations is achieved and then improved by local search. The goal of this recursive process is to intensify the search around current neighborhood structure.

The rest of this paper is organized as follows: Detailed description about ACO algorithm in terms of exploration and exploitation mechanisms with more highlights on hybridization with local search is given in Section 2. Section 3 presents the proposed algorithm, namely reactive max-min ant system with recursive local search algorithm ($RMMAS_{RLS}$), to solve the drawbacks of ACO variants when coupled with local searches. Two new operations to overcome the drawbacks are also described. The experimental design and results are presented in Section 4 and Section 5 respectively while the conclusion is drawn in Section 6.

2. The exploration and exploitation in ACO

ACO is a general purpose optimization framework for problem solving. It takes the inspiration from the foraging behavior of real ants. Ants, in their continuous journey searching for food, mark chemical paths to be followed by other ant foragers of the colony. This type of indirect communication between ants has been described by a mathematical model. It was the result of some experiments called double bridge experiments. When the ant after some explorations finds the food it returns back to the nest using one of the branches of the bridge. By laying and following a chemical instance called pheromone, after time, ants will find the shortest path between food and nest. The natural optimization process of following the pheromone has been modelled computationally into a solution construction model called the state transition rule:

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

Where τ_{ij} is the pheromone value adjusted by the parameter α and μ_{ij} is the pre-heuristic value, which is given by: 1/distance (i, j) and μ is adjusted by the parameter β. The set $N\ (sp)$ represents the set of untraversed edges by ant (k). After generations of ants construct their solutions, the updating phase starts as follow.

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

Where the $\Delta\tau_{ij}^{k}$ is calculated by:

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_{k} & if\ the\ edge\ (i,j)\ \in ant\ (k)'s\ tour \\ 0 & otherwise \end{cases} \quad (3)$$

Where Q is a constant and L_k is the cost of the tour constructed by ant (k) while parameter $\rho \in [0, 1]$ is a pheromone trail decay coefficient. The first model is the analogy of laying pheromone on the ground and the second one is of evaporating pheromone by time. These distinctive models were the main dynamics of the first ant algorithm, namely ant system (AS) (Dorigo, Maniezzo, & Colorni, 1996).

Although the original AS algorithm achieved encouraging results for the TSP problem, it was later found to be inferior to state-of-the-art algorithms for the TSP as well as for other CO problems. The unbalanced designs of exploration and exploitation mechanisms lead to stagnation problems when all the searching agents (i.e. ants) follow the same path (Dorigo & Stützle, 2010). For the purpose of improving the exploration and exploitation behavior, several AS variants have been developed under unified form that is the ant colony optimization framework. Exploration and exploitation are two contrary but complementary processes that are essential for any successful search (Crepinsek, Liu, & Mernik, 2013). Exploration refers to the probing of unvisited regions within the search space, while exploitation refers to the search in the neighborhood structure of high quality good solutions. If exploration takes precedence, the algorithm will explore unproductive areas of the search space before reaching a solution; if exploitation is too strong, the algorithm may converge prematurely and produce a poor result (Solnon, 2010). The substantial difference among ACO algorithms is in the way that they manage the exploration and exploitation balance. The work in exploration and exploitation mechanisms in ACO is divided in to three parts. These are the pheromone memory management, the parameterization and hybridization.

In the first part, several mechanisms can be highlighted such as the elitism in elitist ant system (EAS), trail learning in ant colony system (ACS) and ant based Q-learning (Ant-Q), ranking in rank-based ant system (RAS), trail bounding in MMAS and subtract pheromone in best-worst ant system (BWAS) (Dorigo & Stützle, 2010). More strategies have been stemmed from other computational fields such as such the idea of lower bounds (LB) used in the development of beam search-based ACO (BACO). It is reliable on the completion of a partial solution, which is derived from branch-and-bound. Another approach for memory management relies on the interaction between several pheromone memories. This approach harnessed in developing successful methods for solving multi-objective problems (López-Ibáñez & Stützle, 2012). An advanced memory was added in the so-called population-based ACO (PACO) approach. It has been shown that PACO with restart mechanism was competitive to the standard ACO approach. PACO relies on an auxiliary memory called population for deriving new pheromone updating models (Oliveira, Stützle, Roli, & Dorigo, 2011). The amount pheromone added/dropped relies on the size of the memory, which denoted. This approach contributes in faster pheromone updates for ACO and motivates the invention of more advanced features.

In the second part, the exploration and exploitation is maintained by the parameterization, where the parameters' values of ACO algorithm are changed in an on-line or off-line way. The off-line techniques are either trial-and-error scheme or machine learning scheme. Both schemes are applied before running the algorithm. For the first scheme, it required an experience about the fitness landscape of the CO problem under hand. The lack of this experience in advance is the main obstacle in applying this scheme. For the second scheme, machine learning mechanisms used to propose a suitable parameter setting for each CO instance. These solutions consumes more time before running the algorithm (Pellegrini, Stützle, & Birattari, 2010).The on-line techniques are proposals to change the parameters' values during the run. This may increase the complexity and the burden of development of the algorithm. The lack of successful methodologies for applying

such techniques in ACO justified its poor performance when it is applied to classical CO problem such as TSP or QAP.

In the third part, the combinations between basic techniques for pheromone management (Dorigo & Stützle, 2010) with local search algorithms (Blum et al., 2011) are the most successful approach for improving the quality of solutions produced by ACO algorithm. Although ACO can be applied without coupling with local search procedures, very often its solutions' quality is greatly improved if it is extended to include it. The first step in applying local search is to determine the type of neighborhood. One common way is via k-exchange moves that exchange a set of k components of a solution with a different set of k components. Cheng and Mao (2007) introduced two local heuristics to direct ants during the solution construction. Yanga, Shia, Marcheseb, and Liang (2008) combined the ACO with a mutation and 2-Opt heuristic. Uğur and Aydin (2009) developed a web-based interactive tool to simulate the behavior of several ant colony optimization algorithms with 2-Opt and 2.5-Opt local search procedures. However, the simulation focused on visualizing the optimization process rather than the quality of solutions. Liu (Liu, 2010) applied heap sort algorithm for 2.5-Opt local search and combined them with ACO. Chena and Chiena (2011) hybridized ACO, particle swarm optimization (PSO) and genetic algorithm (GA) with simulated annealing (SA) local search. Gambardella et al. (2012) enhanced the framework of ant colony system when coupled with strong local searches by adding new speedup routines. The definition of strong local search formalized the ability of local search to improve the quality of solutions produced by any search method autonomously and efficiently.

3. The proposed algorithm

This section details the main contributions of this paper. Its start with defines the main memory schemes that will be used for transferring neighborhood structures over iterations. With reference to the general framework of ACO, the probabilistic construction and update pheromone are the main exploration and exploitation components. On the other hand the local search became essential component in the definition of ACO as an aggressive exploitation component. It has the ability to bring the best so far solutions produced by artificial ants to its local minimum. In the presence of local search, the role of the probabilistic construction and update pheromone components is less prominent, together with the limitation that the local search itself in transferring the neighborhood structure over iterations. The local search is applied after artificial ants construct their solutions. In the next iteration, new solutions are generated and new local search is applied. In particular, two operations to overcome the problem are: The first one regards the probabilistic construction and pheromone update component and the second the integration with local search itself.

3.1. Defining the memory schemes

The proposed enhancements rely mainly on utilizing advanced memories, i.e. component-based and population-based memories, to record the features of the neighborhood structures drawn during current search. The first memory scheme, namely components-based memory (CbM) is associated with reactive heuristics (RH) in the context of {0, 1} values. The RHs denote whether the solution component is significant or not. A threshold value ϵ is used to characterize the value of RH. It sets to $1/\rho * C^{nn}$ where ρ is the evaporation rate and C^{nn} is the heuristic value. In each iteration and within the evaporation model of pheromone update, the components of each solution are tested whether it is being decreased below the threshold to be marked as {1} in the corresponding RH, i.e. rh_{ij}. This process will be iterated until a state of convergence is happened. The average λ-branching factor and acceptance criteria described in (Stützle, 1999) are used to determine the convergence phase. This active learning process is highly dependent on the evaporation rate (ρ) that is defined in equation (2). Hence, the values of RHs are derived based on the following formula.

$$f\left(RH,\ \varepsilon, \tau_{ij}\right) = \begin{cases} rh_{ij} \leftarrow 1 & if\, \tau_{ij} < \varepsilon \\ rh_{ij} \leftarrow 0 & if\, \tau_{ij} \geq \varepsilon \end{cases} \tag{4}$$

Where the τ_{ij} the pheromone trail associated with the solution components. After convergence, the values of RH will be exploited as a reactive heuristics to guide the artificial ants in their probabilistic construction process as follows:

$$p_{ij}^{k} = \begin{cases} \frac{\tau_{ij}^{\alpha}.\mu_{ij}^{\beta}.rh_{ij}}{\sum_{\epsilon_{il}\in N\left(s^{P}\right)} \tau_{il}^{\alpha}.\mu_{il}^{\beta}.rh_{il}} & if\ c_{il} \in N\left(S^{P}\right) \\ 0 \quad otherwise \end{cases} \tag{5}$$

(5)

The second memory scheme, namely population-based memory (PbM) is associated with the current population rather than the component of each solution. The main role of PbM is to keep a small population P = {P1,...Pk} of just improved solutions by the so called recursive local search (RLS). The value of k represents the size of memory P and it sets to small value to minimize the computations. After the new solutions have been generated, the population P is updated. If the current local search is succeeded in improving the quality of solution it will be used in P update otherwise, a recursive way of updating is conducted, i.e. RLS. It activated when the typical local search is failed in improving quality of solutions. In RLS, just dropped solutions from P are recursively used in updating. The k-Opt local search algorithms are suitable for hybridization with ACO due to their ability to improve the quality of solutions. The local search manipulated relies on examining the neighbourhood structures with the 2-Opt strategy. For TSP, the 3-Opt local search with *don't look bit* and *nearest neighborhood* data structures (Dorigo & Stützle, 2004) is utilized. For QAP, 3-Opt with the first-improvement strategy is used to hasten the expensive evaluation function property. Therefore, the difference between objective functions before and after local search, i.e. $\Delta f = f(s_{gb}) - f(s'_{gb})$, is hastened. The difference between the local search procedures between TSP and QAP is the speedup techniques where in QAP the second examination is simply cancelled when the first one does not improve the solution (s_{gb}).

3.2. The implementation of RMMAS$_{RLS}$

In this section, an enhanced variant of MMAS denoted as RMMAS$_{RLS}$ is developed after adding the above mentioned algorithmic components, i.e. the reactive heuristics and recursive local search as depicted in Figure 1. Before the algorithm starts solving the CO problem, the CbM and PbM memory structures are initialized. The RH memory records the arcs below the predefined threshold (see Subsection 3.1) of pheromone trails so that the unexplored regions in the current search are shifted to the next search. The ability of ants to remember their previous search influences their future

```
Algorithm: RMMASRLS
  InitializeParameters ()
  Initialize_T_Memory ()
  Initialize_RH_Memory () // CbM scheme
  Initialize_P_Memory () // PbM scheme
  while (not terminate) do // termination condition is fixed to 10 sec in TSP and varied as in Tables 1 & 2 in QAP
    for k := 1 to m do
        if (no stagnation) do
              ConstructSolutions (T, C)
        else
             ReactiveRestart (Sgb, Sk, History)
           ConstructSolutions (T, C, RH)
        end-else
        end-if
        Sib ← argmin{f(Sk | k:=1 to m )}
        if (f(Sib) < f(Sgb))
           Sgb ← argmin{f(Sgb), f(Sib)}
    S'gb ← LocalSearch(Sib)
    Sgb ← argmin{f(Sib), f(S'ib)}
          Add (Sgb)
            if (P = |P|)
            Sob ← Drop ()
    S'ob ← RLS (Sob)
    Sob ← argmin{f(Sob), f(S'ob)}
        else
            if(f(Sob) < f(Sgb))
       Add(Sob)
    else
             Add(Sib)
        Evaporate (RH, T, τmin)
        DepositPheromone (T, Ssb)
    end-for
  end-while
end-algorithm
```

Figure 1. The Pseudocode for RMMAS$_{RLS}$ Algorithm.

decisions according to the new decision model (see Equation 5). The RH's heuristics is beneficial to solve the CO problems even those depends on the pre-heuristic (e.g. distances among cities in TSP). In the proposed model, ants for each decision, will utilize three sources of information: τ_{ij}^{α}, μ_{ij}^{β} and rh_{ij}. Using RH's heuristics, the decision process will be more informative. 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. The pre-heuristic information about search space is becoming ineffectual in the decision of ants. The RH memory stays not active during this time of optimization progress unless the stagnation is indicated by the feedback measures. Once the search is stagnating the search will be is restarted, together with reactivating RH memory and utilizing the new construction solution model.

At each generation of solutions for CO, one of three best solutions is chosen to be added to the P memory: The best-iteration solution, the best-so-far solution or the old-best solution. The first and second solutions are typical choices in other ACO algorithms, while the third one is new. For the management of added/dropped solutions in P memory, a simple first-in-first-out (FIFO)-strategy is adopted. Once a solution is dropped, it is recursively re-added back to P if its quality is improved by local search. This is the so-called recursive local search (RLS). In this way, the neighborhood structure treated by the stochastic local search is explored carefully and the exploitation process is treated more completely.

4. Experimental design

The experiments conducted are divided into three parts: The comparison with ACO algorithms, the application to QAP and the statistical significance of the reported performance improvement for TSP and QAP. The first part of results focuses on the behavior of the proposed algorithm for solving TSP comparing with other ACO variants. The parameter settings for the ACO variants are selected from their literature 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, the value of τ_0 was not exactly defined. Hence it is set to $1/\rho * C^{nn}$. ACO variants are tested with local search 3-opt. The second part of experiments focus on the application of the proposed algorithm to QAP problem. The execution times are proportional to the size and the structure of the QAP instance. These time's durations are for short and long executions. For standard MMAS, the parameter setting is selected from the literature as follows: m = 5; α = 1; ρ = 0.8 and the exploration/exploitation parameter q_0 is equal to 0.5. The evaluation metrics are the average and standard deviation for the quality of solutions of the ten independent runs. In the third part, the non-parametric statistical test Wilcoxon was used to verify the significance of the improvement in quality. Wilcoxon signed-ranks test is based on the positive and negative ranks of the mean of the quality of solutions matric (Derrac, García, Molina, & Herrera, 2011). The test performed with 0.05 significance level and one-tailed hypothesis.

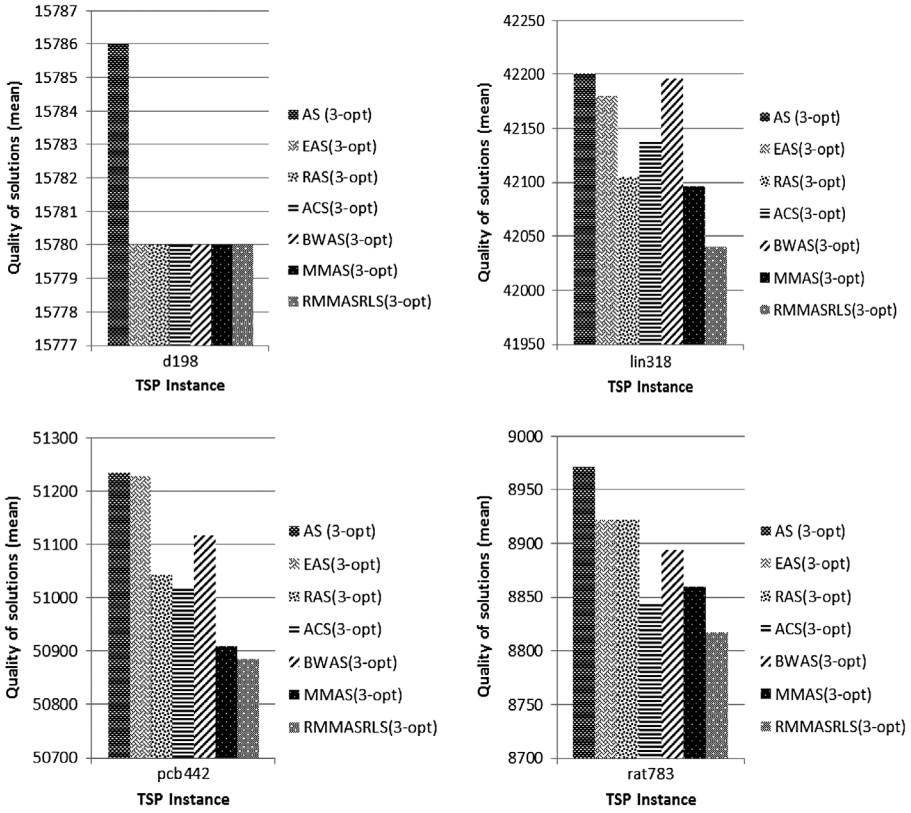

Figure 2. The Performance of $RMMAS_{RLS}$ Versus other ACO Variants for Solving Small and Medium Sizes of TSP Problems.

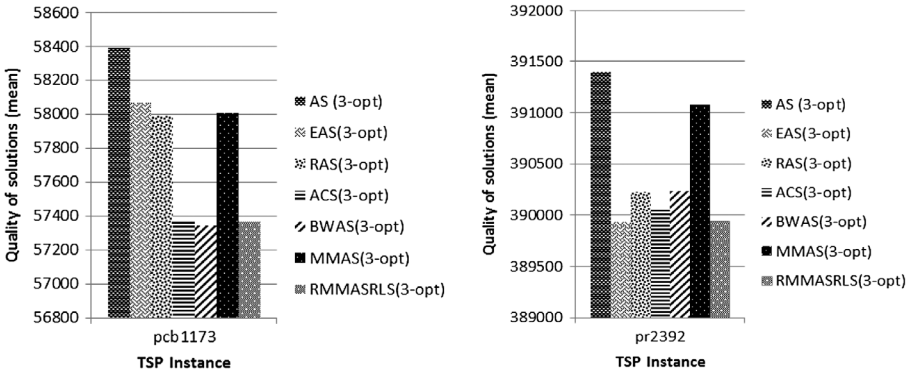

Figure 3. The Performance of $RMMAS_{RLS}$ Versus other ACO Variants for Solving Large Size of TSP Problems.

The QAP and TSP instances are selected from TSPLIB (Reinelt, 1991) and QAPLIB (Burkard, Cela, Karisch, & Rendl, 1997) repositories respectively. For QAP case, selected instances were bur26a, bur26b, bur26c, bur26d, bur26e, bur26f, bur26g, bur26h, chr25a, els19, kra30a, kra30b, tai20b, tai25b, tai30b, tai35b, tai40b, tai50b, tai60b and tai80b while for the TSP case they were st70, Eil76, pr76, gr96, rat99, KroA100, KroB100, KroC100, KroE100, rd100, d198, lin318, pcb442, rat783 and pcb1173. The experiments were conducted on Windows 8 64-bit operating system, processor Intel Core i3–3217U with CPU @ 1.80 GHz, RAM 4 GB. The proposed algorithm was coded in C language. The MMAS-TSP were selected based on ACOTSP 3.0 (Dorigo & Stützle, 2004), where it is easily extended to solve QAP. The parameter settings for the proposed algorithm are described as follows: For the TSP, the k is equal to 10; the ants' number (m) is equal to 25; the pheromone intensity (α) is equal to 1 and evaporation rate (ρ) is 0.2. For the QAP, k = 10; m = 5; α = 1; ρ = 0.8 and the exploration/exploitation parameter q_0 is equal to 0.5. Although it is always better to have a well-tuned algorithm, the parameter setting of the proposed algorithm does not tuned to confirm that the improvement is due to the effectivity of the contributed algorithmic components not a matter of parameter tuning.

5. Computational results

In Figures 2 and 3, the y-axis visualizes the quality of solutions measured by the mean of the best solutions found during the 10 runs conducted to solve TSP. The x-axis represents the ACO algorithms (one bar for each algorithm). There are six standard stochastic local search algorithms namely AS, EAS, RAS, ACS, BWAS and MMAS are used in the experiments. The 3-Opt algorithm are coupled with the tested ACO algorithms including $RMMAS_{RLS}$. The experiments covered three sizes of TSP instances; small; medium and large.

In Figure 2, for the small and medium sizes of instances, the performance of the tested ACO algorithms started equally except in AS algorithm was bad due to the early conference

problem (as in *d198.tsp)*. When the size increased (as in *lin318.tsp)* the performance of the tested algorithms starts to be disparate. The previous two experiments showed that the proposed components are beneficial in solving small scale instances for TSP problem and confirmed the outperformance of the proposed algorithm. For the medium size of instances, the outperformance of the proposed algorithm continued. For the TSP instance *pcb442.tsp*, the MMAS and the $RMMAS_{RLS}$ are the best among other ACO variants. The proposed algorithm inherits the exploration and exploitation components of MMAS. For the TSP instance *rat732.tsp*, the ACS and the $RMMAS_{RLS}$ are the best among other ACO variants. Here, the aggressive exploitation of the proposed RLS technique is dominates the standard local search coupled with ACS. The proposed components are still beneficial in solving this size of instances for TSP problem.

For the large size of instances, the proposed algorithm is still competitive except in pcb1173.tsp instance the results were competitive but not better than BWAS. However, the gap is not such big so the ACS and $RMMAS_{RLS}$ are still showing the same performance. It is worth mentioning that the proposed algorithm is not well-tuned while the parameter setting of all other ACO variants is optimized as shown in the literature. The results showed that the proposed algorithm and the EAS are the best where the exploitation is the dominant process in large instances. Hence, the proposed algorithm also showed good performance with large TSP instances. As overall result for this part of experiments, the performance of the proposed algorithm is good in solving all sizes of TSP even with tight run time. It outperformed other ACO variants for solving all TSP instances except *pcb1173.tsp*. It can be seen from the results that coupling local search procedures with ACO algorithms did not affect the outperformance of the proposed algorithm. That is because of the aggressive exploitative contributed by RLS technique throughout the searching period with the ability to turn to exploration contributed by reactive heuristics (see Figure 3).

In the subsequent subsections the results of the second part of experiments is presented. The results of the application of

Table 1. Performance of $RMMAS_{RLS}$ Algorithm on QAP Instances (Short run).

			MMAS-QAP			$RMMAS_{RLS}$-QAP		
QAPinstance	Best known	Sec.	*Mean*	*SD*	*Best*	*Mean*	*SD*	*Best*
bur26a	5,426,670	8	5,427,097	636	5,426,670	**5,426,670**	**0**	5,426,670
bur26b	3,817,852	8	3,817,935	73	3,817,852	**3,817,852**	**0**	3,817,852
bur26c	5,426,795	8	5,426,893	107	5,426,795	**5,426,795**	**0**	5,426,795
bur26d	3,821,225	8	3,821,255	48	3,821,225	3,821,225	**0**	3,821,225
bur26e	5,386,879	8	5,387,074	185	5,386,879	**5,386,879**	**0**	5,386,879
bur26f	3,782,044	8	3,782,048	6	3,782,044	**3,782,044**	**0**	3,782,044
bur26 g	10,117,172	8	10,117,324	182	10,117,172	**10,117,172**	**0**	10,117,172
bur26 h	7,098,658	8	7,098,708	103	7,098,658	**7,098,658**	**0**	7,098,658
chr25a	3,796	4	4,562	172	4,304	**4,177**	**99**	**3,984**
els19	17,212,548	2	17,241,610	43,635	17,212,548	**17,212,548**	**0**	17,212,548
kra30a	88,900	8	95,609	224	95,145	**94,372**	**157**	**94,130**
kra30b	91,420	9	92,298	217	91,900	**91,523**	**120**	**91,420**
tai20b	122,455,319	3	122,667,105	172,642	122,455,319	**122,455,319**	**0**	122,455,319
tai25b	344,355,646	5	345,428,471	762,772	344,653,810	**344,379,559**	**75,620**	**344,355,646**
tai30b	637,117,113	9	638,804,383	580,656	637,743,822	**637,218,046**	**258,852**	**637,117,113**
tai35b	283,315,445	15	284,997,173	371,182	284,180,375	**283,768,905**	**241,686**	**283,315,445**
tai40b	637,250,948	24	639,646,179	677,314	638,452,551	**637,375,646**	**133,920**	**637,250,948**
tai50b	458,821,517	50	461,287,056	853,848	459,959,918	**459,293,938**	**126,779**	**459,121,468**
tai60b	608,215,054	90	612,310,940	960,895	611,081,614	**608,922,672**	**297,495**	**608,387,539**
tai80b	818,415,043	225	828,968,489	3,493,073	822,936,304	**822,384,964**	**1,731,411**	**820,317,326**

Note: Bold values indicate that the quality of solutions provide by the proposed algorithm is better than the standard algorithm as shown by the lower ***mean***, standard deviations (***SD***) and ***best*** quality.

Table 2. Performance of $RMMAS_{RLS}$ Algorithm on QAP Instances (Long-run).

			MMAS-QAP			$RMMAS_{RLS}$-QAP		
QAPinstance	Best known	Sec.	*Mean*	*SD*	*Best*	*Mean*	*SD*	*Best*
bur26a	5,426,670	50	5,426,670	0	**5,426,670**	**5,426,670**	0	5,426,670
bur26b	3,817,852	50	3,817,853	4	3,817,852	**3,817,852**	**0**	**3,817,852**
bur26c	5,426,795	50	5,426,796	2	5,426,795	**5,426,795**	**0**	**5,426,795**
bur26d	3,821,225	50	3,821,225	0	**3,821,225**	**3,821,225**	0	3,821,225
bur26e	5,386,879	50	5,386,879	0	**5,386,879**	**5,386,879**	0	5,386,879
bur26f	3,782,044	50	3,782,044	0	**3,782,044**	**3,782,044**	0	3,782,044
bur26 g	10,117,172	50	10,117,172	0	**10,117,172**	**10,117,172**	0	10,117,172
bur26 h	7,098,658	50	7,098,658	0	7,098,658	7,098,658	0	7,098,658
chr25a	3,796	40	4,154	**116**	3,946	**4,042**	144	**3,796**
els19	17,212,548	20	17,212,548	0	17,212,548	**17,212,548**	0	17,212,548
kra30a	88,900	76	94,588	151	94,340	**94,239**	**148**	**93,930**
kra30b	91,420	86	91,517	88	91,420	**91,434**	**29**	**91,420**
tai20b	122,455,319	27	122,455,319	0	122,455,319	**122,455,319**	0	122,455,319
tai25b	344,355,646	50	344,496,014	122,804	344,355,646	**344,355,646**	**0**	**344,355,646**
tai30b	637,117,113	90	637,612,929	440,084	637,152,585	**637,128,942**	**11,091**	**637,117,113**
tai35b	283,315,445	147	284,231,012	**101,855**	284,027,477	**283,378,972**	134,301	**283,315,445**
tai40b	637,250,948	240	638,153,448	381,309	637,598,806	**637,259,823**	**19,516**	**637,250,948**
tai50b	458,821,517	480	460,204,146	349,142	459,529,895	**459,036,877**	**59,112**	**458,923,553**
tai60b	608,215,054	855	610,393,364	462,570	609,780,832	**608,563,150**	**104,995**	**608,387,539**

Note: Bold values indicate that the quality of solutions provide by the proposed algorithm is better than the standard algorithm as shown by the lower ***mean***, standard deviations (***SD***) and ***best*** quality.

Table 3. Wilcoxon Signed-ranks Statistical Test of the Performance of $RMMAS_{RLS}$ Algorithm.

RMMAS vs.	*R+*	*R-*	*p-value*
For TSP:			
AS	**136**	0	**0.00022**
EAS	**119**	1	**0.0004**
RAS	**120**	0	**0.00032**
ACS	**96**	9	**0.00317**
BWAS	**119**	1	**0.0004**
For QAP:			
MMAS (short)	**210**	0	**8.00E-05**
MMAS (long)	**66**	0	**0.00169**

Note: Bold values represent the number of times in which the proposed algorithm has outperformed (***+R***) while the ***P-values*** indicate to which level the results were statistically significant.

$RMMAS_{RLS}$ to the solution of the difficult combinatorial problem, i.e. QAP, seem to indicate that combining advanced local search mechanisms with the best performing ACO methods can give rise to powerful optimization algorithms. It is focuses on the type of fitness landscape traversed. That is done by the application of the algorithm $RMMAS_{RLS}$ to twenty (20) QAP instances in different sizes. In Tables 1 and 2, it is clear that the $RMMAS_{RLS}$ is the best performing than the original MMAS for the short and long runs. The results showed significant improvement in the quality of solutions produced by the proposed algorithm against the original MMAS-QAP algorithm. For some QAP instances of the type *bur26x.qap*, the original MMAS-QAP algorithm can produce high quality solutions in long run but the overall performance over the ten (10) runs showed that the proposed algorithm is better. On the other side, for some instances such as *chr25a.qap* and *tai35b.qap* the MMAS-QAP is able to compete in the overall performance as the medians for such solutions is same.

Table 3 presents the statistical results of applying RMMAS to TSP and QAP with comparison to AS, EAS, RAS, ACS and BWAS for the TSP and compared to MMAS for the QAP. Wilcoxon signed-ranks statistical test showed that the search of RMMAS is more robust as it outperforms the said ACO variants in the number of ranks for mean of the quality of solutions. In the comparison of means, the RMMAS algorithm collected (136 of 136), (119 of 120), (120 of 120), (96 of 105) and (119 of 120) ranks against AS, EAS, RAS, ACS and BWAS respectively for the TSP and (220 of 220) and (66 of 66) for short and long runs of MMAS respectively. The result was significant at $p \leq 0.05$ where p-values were equal to (0.00022), (0.0004), (0.00032), (0.00317) and (0.004) for the RMMAS comparing to AS, EAS, RAS, ACS and BWAS respectively for TSP. The p-values were (800E-05) and (0.00169) for short and long runs of RMMAS comparing to MMAS respectively for QAP.

6. Conclusion

The proposed reactive heuristics and recursive local search technique are based on a combination between local search procedures, auxiliary memories and the distributed computation of ants in ACO algorithm. The resulting algorithm, i.e. $RMMAS_{RLS}$ has been used to tackle the problem of premature exploitation. These contributions are empirically and statistically confirmed by the results with fair comparison. Future work for this research is the application of advanced feedback mechanisms which will focus on similarity and diversity concepts due to the correlation with exploitation and exploration. Online parameter adaptation is another direction to automate the exploration/exploitation balance of the proposed algorithm.

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Rafid Sagban holds a Bachelor in Computer Sciences from University of Babylon in 1999. His Master's degree and Ph.D. in I.T are both from Universiti Utara Malaysia in 2015. Rafid's experience includes over 13 years in the Academia/Industry involving algorithm analysis, swarm intelligence, and web development.

Ku Ruhana Ku-Mahamud holds a Bachelor in Mathematical Sciences and a Master's degree in Computing, both from Bradford University, UK in 1983 and 1986 respectively. Her Ph.D. in Computer Science was obtained from Universiti Pertanian Malaysia in 1994. Her research interests include computer systems modeling, ant colony optimization and intelligent agent.

Muhamad Shahbani Abu Bakar received the B.Sc. Computer Science and M.Sc (IT) from, Universiti Putra Malaysia and Universiti Utara Malaysia in 1990 and 2001, respectively. His Ph.D. in Software Engineering was from Universiti Utara Malaysia in 2012. His academic experience includes software engineering, web & mobile applications, data warehouse and business intelligence.

References

Battiti, R., Brunato, M., & Mascia, F. (2008). *Reactive search and intelligent optimization*. U.S.A.: Springer.

Blum, C., Puchinger, J., Raidl, G.R., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing, 11*, 4135–4151.

Burkard, R., Cela, E., Karisch, S.E., & Rendl, F. (1997). Benchmark-QAPLIB : A quadratic assignment problem library. *Journal of Global Optimization, 10*, 391–403. Retrieved from http://www.opt.math.tu-graz.ac.at/qaplib/

Mohan, B.C., & Baskaran, R. (2012). A survey : Ant colony optimization based recent research and implementation on several engineering domain. *Expert Systems with Applications, 39*, 4618–4627. doi:http://dx.doi.org/10.1016/j.eswa.2011.09.076.

Chena, S.-M., & Chiena, C.-Y. (2011). Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. *Expert Systems with Applications, 38*, 14439–14450.

Cheng, C.-B., & Mao, C.-P. (2007). A modified ant colony system for solving the travelling salesman problem with time windows. *Mathematical and Computer Modelling, 46*, 1225–1235.

Crepinsek, M., Liu, S.-H., & Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms : A survey. *ACM Computing Surveys, 45*(3), 1–33. doi:http://dx.doi.org/10.1145/2480741.2480752

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation, 1*, 3–18. doi:http://dx.doi.org/10.1016/j.swevo.2011.02.002

Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. Cambridge, MA, USA: MIT Press.

Dorigo, M., & Stützle, T. (2010). Ant colony optimization: Overview and recent advances. In M. Gendreau & J. Potvin (Eds.), *Handbook of metaheuristics* (Vol. 146, pp. 227–263). New York, USA: Springer, US.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transaction on Systems, 26*, 29–41. doi:http://dx.doi.org/10.1109/3477.484436

Gambardella, L.M., Montemanni, R., & Weyland, D. (2012). Coupling ant colony systems with strong local searches. *European Journal of Operational Research, 220*, 831–843. doi:http://dx.doi.org/10.1016/j.ejor.2012.02.038

Liu, X. (2010). Ant colony optimization based on an improvement local search strategy. *Journal of Computational Information Systems, 6*, 3423–3430.

López-Ibáñez, M., & Stützle, T. (2012). An experimental analysis of design choices of multi-objective ant colony optimization algorithms. In M. López-Ibáñez & T. Stützle (Eds.), *Swarm Intelligence* (Vol. 6, pp. 207–232). Berlin Heidelberg: Springer. doi:http://dx.doi.org/10.1007/s11721-012-0070-7

Oliveira, S., Stützle, T., Roli, A., & Dorigo, M. (2011). A Detailed analysis of the population-based ant colony optimization algorithm for the TSP and the QAP. In Proceedings of GECCO'11: Genetic and evolutionary computation conference (pp. 13–14). doi:http://dx.doi.org/10.1145/2001858.2001866

Pellegrini, P., Stützle, T., & Birattari, M. (2010). *Tuning MAX—MIN ant system with off-line and on-line methods*. Bruxelles: IRIDIA, Institut de Recherches Interdisciplinaires et de D´eveloppements en Intelligence in Universite Libre de Bruxelles.

Reinelt, G. (1991). Benchmark-TSPLIB: A traveling salesman problem library. *ORSA Journal On Computing*. Retrieved from http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Solnon, C. (2010). *Ant colony optimization and constraint programming*. U.S.A.: Weley & Sons Ltd. doi:http://dx.doi.org/10.1002/9781118557563

Stützle, T. (1999). *Local search algorithms for combinatorial problems: Analysis, improvements, and new applications*. Unpublished doctoral dissertation. Technische Universit, Darmstadt, German.

Stützle, T., & Hoos, H.H. (2000). MAX—MIN ant system. *Future Generation Computer Systems, 16*, 889–914.

Stützle, T., Lopez-Ibanez, M., & Dorigo, M. (2010). A concise overview of applications of ant colony. In J.J. Cochran (Ed.), *Wiley Encyclopedia of Operations Research and Management Science* (pp. 896–911). John Wiley & Sons.

Uğur, A., & Aydin, D. (2009). An interactive simulation and analysis software for solving TSP using Ant Colony Optimization algorithms. *Advances in Engineering Software, 40*, 341–349. doi:http://dx.doi.org/10.1016/j.advengsoft.2008.05.004

Wanga, H., Wua, Z., Yanga, X., & Liua, H. (2011). A novel ACO-based multicast path algorithm in hypercube networks. *Intelligent Automation & Soft Computing, 17*, 541–549.

Yanga, J., Shia, X., Marcheseb, M., & Liang, A. (2008). An ant colony optimization method for generalized TSP problem. *Progress in Natural Science, 18*, 1417–1422.