European Journal of Operational Research 254 (2016) 19–28

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Continuous Optimization

Eidetic Wolf Search Algorithm with a global memory structure

Simon Fong[a], Suash Deb[b], Thomas Hanne[c,*], Jinyan (Leo) Li[d]

[a] Department of Computer and Information Science, University of Macau, Macau
[b] Department of Computer Science and Engineering, Cambridge Institute of Technology, India
[c] Institute for Information Systems, University of Applied Sciences and Arts Northwestern Switzerland, Riggenbachstr. 16, 4600 Olten, Switzerland
[d] Department of Computer and Information Science, University of Macau, Taipa, Macau

ARTICLE INFO

Article history:
Received 7 August 2014
Accepted 23 March 2016
Available online 14 April 2016

Keywords:
Metaheuristics
Wolf Search Algorithm
Global memory structure
Ant Colony Optimization
Particle Swarm Optimization

ABSTRACT

A recently proposed metaheuristics called Wolf Search Algorithm (WSA) has demonstrated its efficacy for various hard-to-solve optimization problems. In this paper, an improved version of WSA namely Eidetic-WSA with a global memory structure (GMS) or just eWSA is presented. eWSA makes use of GMS for improving its search for the optimal fitness value by preventing mediocre visited places in the search space to be visited again in future iterations. Inherited from swarm intelligence, search agents in eWSA and the traditional WSA merge into an optimal solution although the agents behave and make decisions autonomously. Heuristic information gathered from collective memory of the swarm search agents is stored in GMS. The heuristics eventually leads to faster convergence and improved optimal fitness. The concept is similar to a hybrid metaheuristics based on WSA and Tabu Search. eWSA is tested with seven standard optimization functions rigorously. In particular, eWSA is compared with two state-of-the-art metaheuristics, Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO). eWSA shares some similarity with both approaches with respect to directed-random search. The similarity with ACO is, however, stronger as ACO uses pheromones as global information references that allow a balance between using previous knowledge and exploring new solutions. Under comparable experimental settings (identical population size and number of generations) eWSA is shown to outperform both ACO and PSO with statistical significance. When dedicating the same computation time, only ACO can be outperformed due to a comparably long run time per iteration of eWSA.

1. Introduction

In mathematical science it is generally known that there is no efficient algorithm to solve NP-complete optimization problems, thereby motivating researchers to innovate stochastic algorithms for generating approximate instead of deterministic solutions. Also for difficult continuous optimization problems with multiple local optima, nondifferentiable objective functions, or with challenging constraints, stochastic algorithms have often shown to be superior to more traditional optimization methods. Swarm intelligence algorithms have lately emerged in computational science. These showed good results in various applications of optimization domains, such as supply chain logistics (Chiang, Che, & Cui, 2014), flowshop scheduling (Zhang & Wu, 2014) and mechanical design (Cheng & Lin, 2014) by Particle Swarm Optimization algorithms; RFID network optimization (Ma, Chen, Hu, & Zhu, 2014), image processing (Charansiriphaisan, Chiewchanwattana, & Sunat, 2013), and cast scheduling (Pan, 2016) by Artificial Bee Colony Algorithm; computer vision (Xiao, Xia, Zhu, Huang, & Xie, 2013), power channel optimization (Lin, Tsai, & Lee, 2013), routing in mobile networks (Rupérez Cañas, Sandoval Orozco, García Villalba, & Hong, 2013), industrial layout problems (Hani, Amodeo, Yalaoui, & Chen, 2007), pattern matching (Sreeja & Sankar, 2015), and vehicle routing (Schyns, 2015) by Ant Colony Optimization (ACO). One of the most prominent representatives of swarm intelligence is Particle Swarm Optimization (PSO) (Eberhart & Kennedy, 1995) which has been used successfully in a broad range of applications (Poli, Kennedy, & Blackwell, 2007), e.g. for flowshop scheduling problems (Tseng & Liao, 2008) or in neural network-based applications (Sermpinis, Theofilatos, Karathanasopoulos, Georgopoulos, & Dunis, 2013). PSO is also popular because of its simplicity and applicability to various types of optimization problems. Another very popular algorithm in this group of methods are ACO algorithms which have been shown to be quite successful for finding optimal or almost optimal solutions for difficult optimization problems (Dorigo, 2001; Liao, Stützle, de Oca, & Dorigo, 2014; Socha & Dorigo, 2008). More recently emerging swarm intelligence

* Corresponding author. Tel.: +41 629572292.
E-mail addresses: ccfong@umac.mo (S. Fong), suashdeb@gmail.com (S. Deb), thomas.hanne@fhnw.ch, thomas.hanne@gmail.com (T. Hanne), ljyjy86@hotmail.com (J. Li).

http://dx.doi.org/10.1016/j.ejor.2016.03.043
0377-2217/

algorithms are, for instance, the Bat Algorithm (Yang, Deb, & Fong, 2014b), Cuckoo Search (Yang & Deb, 2013), the Krill Herd algorithm (Wang, Deb, & Cui, 2015), and Monarch Butterfly Optimization (Wang, Gandomi, Alavi, & Deb, 2015).

While ACO is one of the most representative algorithms that utilize stigmergy for building new solutions stochastically, the Wolf Search Algorithm (WSA) (Tang, Fong, Yang, & Deb, 2012) can be potentially improved by stigmergy. To guide the search towards promising solutions, ACO retains the heuristic information from past generations as pheromone trails. The strength of the pheromone trail is reinforced iteratively depending on the quality of the solutions found by the ants. The ants moisten the trails which are relatively more favorable with fresh pheromones in the direction of the highest pheromone concentration. With the increased probability of the strongest pheromone trail by which the ants travel, the ants eventually will converge to the most frequently used trail as the fittest solution. Other swarm algorithms such as PSO have been enhanced by using memory from the path of visited solutions as well (Yin, Glover, Laguna, & Zhu, 2010).

Using the same concept of artificial stigmergy which is a mechanism of communication among the search agents by modifying the environment, a modified version of WSA called Eidetic-WSA (eWSA) is proposed in this paper. eWSA contributes to fast convergence and possibly yields better solutions than the original WSA. The objective of this paper is to investigate the efficacy of eWSA in comparison particularly to ACO and PSO which are both popular and proven successful in many applications. The rest of the paper is organized in this way: Section 2 reviews the background of incorporating external memory into metaheuristic search agents. The principles of the proposed eWSA are described in Section 3. The experiments which are designed to validate eWSA and compare eWSA versus ACO and PSO are presented in Section 4. The results are then discussed in Section 5. Section 6 concludes the paper.

2. Background

In the research field of metaheuristics and other search strategies, the concept of incremental improvement of the optimal solution by incorporating heuristic information from previous iterations has become prevalent. Technically the heuristic information could be retained by using internal or external memory. For instance, Montgomery and Randall (2003) and Eggermont and Lenaerts (2001) are pioneers who made use of internal memory for retaining the information in the stochastic process. Using external memory, however, for keeping the knowledge from previous iterations is originated in Guntsch and Middendorf (2002). Recently for ACO, extensions of optimization models built with external memory strategies demonstrated improvement in performance as reported in Acan (2004) and Salto, Leguizamón, and Alba (2009). The external memory is used to store the previous solutions from ants that walked the trails with the strongest pheromone. A new segment is hauled out from the trail that is represented by a stored previous solution, by randomly enclosing the starting and ending positions. The search agents then considered the heuristic information from the external memory in addition to the main memory. The quality of the solutions, resulting from the main memory and the external memory are compared respectively. Another significant work (Acan, 2005) extended ACO by randomly comprising partial permutation sequences of the solutions into external memory. It has been shown that the new modification offered solutions superior to the traditional ACO algorithms. Salto et al. (2009) applied the same modification with external memory into ACO, for solving a 2D strip packing problem. The external memory stored partial permutation solutions as well as some heuristic knowledge from the problem, like the layout of the pieces that clue the decision over selecting the preferred partial permutation solutions. This idea is similar to choosing building blocks that seem to be good solutions in genetic algorithms (Goldberg, 1989). The external memory keeps track of good combination of pieces, which enables in finding good quality solutions and speeding up convergence.

There are other precedent successful examples of implementing hybrid metaheuristics with incremental improvement for solving a broad range of combinatorial optimization problems. One of the most considerable cases is the integration of genetic operations into the evolutionary process of ACO (Nada & Al-Salami, 2009). With the enhancement of ant movement inclined towards better solution states, the sub-solutions are accumulated which enables fast convergence to an optimal solution eventually. Specifically for solving bipartite subgraph problems, a memory-based ACO (Wang & Zhao, 2010) was developed where the ants are equipped with memory, remembering the previously found solutions for constructing new solutions. The same authors tested the proposed hybrid algorithm for a travelling salesman problem (TSP), too (Wang, Zhao, & Zhou, 2012). A similar variant of ACO is called Cunning Ants (Tsutsui, 2006) that has shown to work very well on TSP problems by avoiding local optima and allows for fast convergence. The ants are called cunning in a sense of remembering and borrowing some parts of the solutions constructed from previous iterations, instead of fully relying on the pheromone density to generate a new solution. Mavrovouniotis and Yang (2011) implemented and tested a variant of memory-based ACO for solving the dynamic TSP problem, which demonstrated the feasibility of memory-based metaheuristic schemes in cyclic dynamic environments. Goni and Cordero (2009) also applied a similar memory-based ACO on a graph search problem where cleaner robots are searching for piles of marks to clean in a gallery. The problem is based on a TSP formulation by removing the superfluous nodes which may appear as the resolution of the problem progresses, for search speed-up.

3. Our proposed Eidetic-WSA

A new breed of swarm intelligence, called Wolf Search Algorithm (WSA) (Tang et al., 2012), was recently invented. Unlike other swarm-type algorithms, WSA agents swarm to relatively less extent as compared to the other search algorithms, where agents follow a flock towards a single leading agent (hence a singular direction), whereas WSA agents search autonomously. Given the precedent success of ACO with the rise of hybrid heuristics which use solutions from previous iterations in an external memory, it is motivating to transform search agents from WSA to Eidetic-WSA in the hope of gaining the same advantages. There are various similarities between these two metaheuristics. A problem and its possible solutions are represented in a high-dimensional search space, where a comprehensive exploration of the search space is usually prohibited by computational and time constraints. The fundamental and underlying operation of the two algorithms is an iterative loop in which a population of search agents builds and improves the candidate solutions progressively: ants are guided probabilistically by heuristic information such as pheromone trails which are collectively maintained among the population. The pheromone concentration represents the degree of desirability for the ants to undertake. Likewise, wolves can make use of stigmergy such as their footprints that marked their previous trails, for improving their search direction. Footprints are used as clues in addition to their visual distance for finding and merging to their peers which are believed to have been placed at better positions in the search space.

The footprints of the wolves which are analogous to past trails are maintained in a global memory, with the visited places in the search process and their associated fitness values in memory. Instead of memorizing the best position and the highest fitness

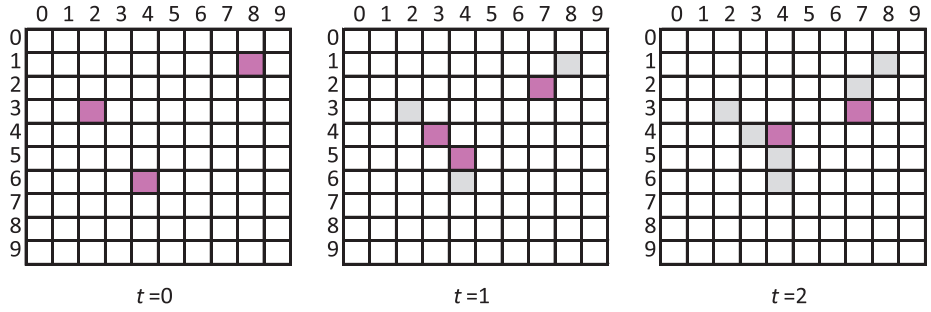

Fig. 1. Snapshot of a WSA movement and visited places to be memorized.

value in each iteration, a tabu list is used to accumulate all the visited places except the best ones from previous iterations. In other words, the relatively less desirable places which have already been visited by the wolves are collectively memorized from iteration to iteration. Thus, the search agents will not waste time and computational resources in going back there again during their random walk. By not considering those tabu places, the search agents are likely to move away from undesirable places and swarm into a better terrain in general. Wolves in this implementation are assumed to have strong photographic memory, remembering every step they took in the search space, especially the undesirable ones (with low fitness values), hence the notation eidetic wolves.

Similar to the original Tabu Search by Glover (1986, 1989), the global memory structure (GMS) maintained by eWSA is a tabu list which is essentially a list of banned solutions in the search space used to filter which solutions to be considered in a new iterative generation for being explored by the swarm search. Assume there is a population of W wolves and in each iteration the GMS includes all the visited places minus the best place. The best place is one that has the highest fitness value in the current generation that naturally should be allowed to visit again for the sake of final convergence. So $(W-1)\times$ *max. no. of iterations* visited places may need to be stored in total, assuming the search space is infinite. In its actual form, the GMS can be a list of limited length built from the old solutions that have been visited in the recent past, depending on the number of past generations that are required to be memorized. The length could be short, medium or long-term depending on how many old solutions are considered as relevant for forthcoming search and on how much memory is to be dedicated for the list. The GMS is implemented as a linked list structure in computer runtime memory for fast indexing and memory efficiency. This implementation could be further improved, e.g. by using a binary search tree or hash table which have better time complexity than using a linked list (amortized O(1) lookup time for the hash table compared to O(u) in case of the linked list). As the number of items in the linked list (u) can probably be restricted to values around 20–50 (see Section 5) the specific implementation of the GMS is not of primary interest within our study. Another possibility for implementing the tabu list to be analyzed in future research is the usage of a vector (an array) which should allow a faster access on stored values. This is always possible when the length of the tabu list is limited but probably also for the unlimited case when a maximum utilization can be observed based on experiments (see Section 5).

In Fig. 1, a simple example of the potential visited places to be stored in the GMS is shown, where there are 3 wolves searching the search space and no winning fitness is observed so far. After two iterations in Fig. 1, the GMS may potentially accumulate (2.3), (3.4), (4.5), (4.6), (7.2) and (8.1) from the 2D search space.

Let a general optimization problem be defined as finding $x^{opt} = \text{argmin}_{x\in X} f(x)$ where X is the search space and $f(x)$ is a fitness function measuring the goodness of the solution (objective function). Without loss of generality, we assume that the fitness function is to be minimized. The global optimum represents the best solution x^{opt} that is assumed to exist in the combinatorial problem space that is too large to be completely searched by brute force. Each search agent in WSA represents a candidate solution by its position in the search space, with respect to a given measure of quality defined by a fitness function. A wolf generates a candidate solution $x_w^{candidate}$ where $w \in \{1...W\}$ and W is the total number of wolves, and as long as the stopping criteria are not met, it roams around by a random walk in the hope of finding a better solution (FIND $x^{neighbor} \in$ neighborhood($x^{candidate}$)). The candidate solution is updated with its neighbor if it is better (IF $f(x^{neighbor}) < f(x^{candidate})$ THEN $x^{candidate} = x^{neighbor}$), such that the global optimum reached at the end is $x^{opt} = x^{candidate}$. Together with merging with its peers (in case that they seem to have better fitness), this mechanism forms the activities of the local search which is sometimes denoted as intensification. For diversification, the wolves jump out of their pack afar for diverse exploration. In connection with the movements of the wolves, i.e. during the random walk, grouping and jumping out, the GMS takes effect in prohibiting the wolves from revisiting the former places except those that are shown to yield high fitness values. See Fig. 2 for a pseudo code description of the algorithm.

While the details of the WSA algorithms are reported in Tang et al. (2012), the logics of the eWSA are presented as follows: let each wolf have a visual range with a radius of v. X is a continuous set of possible solutions. Thus, the visual distance for spotting a peer wolf is defined as:

$$d(x_i, x_c) = \left(\sum_{k=1}^{n} |x_{ik} - x_{ck}|^{\lambda} \right)^{1/\lambda} \leq v \qquad (1)$$

where x_i is the current position of the wolf; x_c are all the potential neighboring positions near x_i and the absolute distance between the two positions must be equal to or less than v; and λ is a parameter for the applied metrics, e.g. $\lambda = 2$ in the case of a Euclidean metrics. s is the step size by which a wolf roams at a time in random local search. W is the total number of wolves that search in the search space in parallel. α is the velocity factor for the wolf, determining their speed of movement and merging. $p_a \in [0...1]$ is a user-defined threshold that determines how often a wolf moves far away escaping from its local search proximity when any threat (e.g. human hunter or tiger) is encountered. The objective function $f(x)$ where $x = (x_1, x_2, ..., x_d)^T$ is estimated by an iterative process bounded by stopping criteria, such as the difference of the fitness gain between two generations falls below a predefined threshold or the maximum number of iterations is reached. *Tabu-list* is the implementation of GMS, and *sBest* stores the place among the visited places in each round that has the highest fitness.

The algorithm makes use of three different update routines for the location, Prey_new_food_initiatively, Prey_new_food_passively, and Escape_region. The first two of these routines determine

```
Input:  f: objective function to be minimized
        W: number of wolves
        pa: probability that a wolf does not leave the local proximity
        α: velocity factor determining the speed of wolves
        s: step size factor determining the distance for escaping local regions
        v: radius of visual range
        u: length of the tabu list

Initialize the population of wolves, xi(i=1,2,..,W)
Initialize the GMS, with variables best_loc, loc, Tabu-list ← null
t = 0;
WHILE (t++, not check_stopping_criteria())
  FOR i=1:W  // for each wolf
    xi = Prey_new_food_initiatively(xi, α, v, Tabu-list);
    loc = xi
    FOR j=1:W  // for each wolf
      IF(dist(xi,xj) ≤ v && f(xj)<f(loc)) // other wolf location is better and within the visual range
        loc = xj // replace best solution found before by solution xj
      END-IF
    END-FOR
    IF (xi != loc)
      xi = loc  // xi moves towards loc which is better than xi and within the visual range
    ELSE-IF
      xi = Prey_new_food_passively(xi, α, v, Tabu-list);
    END-IF
    IF(rand() > pa)  // time to jump out of the local proximity
      xi = Escape_region(xi, α, s, Tabu-list); // escape to a new position.
    END-IF
    Best_loc = Update_the_best_location()
    Tabu-list = Update_Tabu_list(all visited locations at t except Best_loc, u)
  END-FOR
END-WHILE
```

Fig. 2. Pseudo-code for the eWSA algorithm.

a new location randomly around the current location within a radius of $\alpha \cdot v$. In the case of Escape_region the new location is determined randomly around the current location within a radius of $\alpha \cdot s$ with s being significantly larger than v. In each of these procedures, a new generated location is compared with the locations from the tabu list, and in case of being in the tabu list, new locations are generated until nonvisited solutions are found. Readers are referred to Tang et al. (2012) for further details of the WSA operations.

With respect to time complexity of the algorithm, we can note that each iteration requires time quadratically dependent on the number of wolves multiplied by the length of the tabu list. The tabu list is used (checked and updated) up to four times per iteration. If the length of the tabu list is restricted this would not lead to an increase of time complexity in big O notation. If it is unlimited, the worst case complexity per iteration could lead to time requirements proportional to W^3 multiplied by the number of iterations which is, however, not an issue being observed in our experiments (see Sections 4 and 5).

4. Numerical experiments

In order to validate the performance of eWSA, in comparison to WSA (with an empty tabu list), its counterpart ACO and PSO as another state-of-the-art swarm algorithm, the algorithms are put under test with a collection of seven popular optimization functions. In all cases, these test function are bound constrained and have mostly numerous local minima and a certain global minimum (see the Appendix for further details). Namely, these functions are Schaffer's, Sphere, Rosenbrock, Griewank, Michalewicz, Rastrigin, and Moved axis; and they are defined in Appendix. For example, one of the seven optimization functions, called Rastrigin's function (Törn & Zilinskas, 1989) is a non-convex function for testing a metaheuristic algorithm. Rastrigin's function is a highly multimodal optimization problem which provides particular difficulties to an optimization method. The function is defined as follows:

$$f(x) = 10n + \sum_{i=1}^{d} (x_i^2 - 10\cos(2\pi x_i)) \tag{2}$$

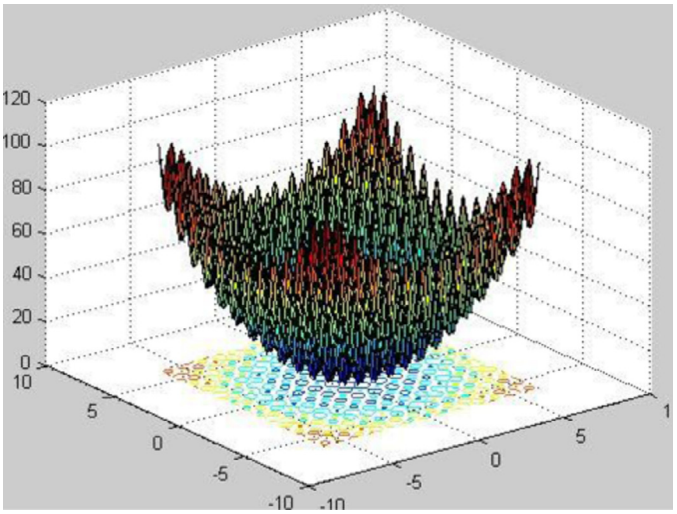

Fig. 3. The 3D visualization of Rastrigin's function.

where $x_i \in [-5.12, 5.12]$. Rastrigin's function in its 3D manifestation is visualized in Fig. 3. The global minimum of Rastrigin's function is $f(\cdot) = 0$ at $(0, \ldots, 0)$. It can be seen that the function is filled with local minima all over the search space.

The test problems can be defined for various dimensions d of the search space; for the first two series of experiments and a fourth series we, however, stick to low-dimensional problems ($d = 2$) whereas higher dimensional problems are considered in the third series of experiments.

To make the problem more challenging for WSA and eWSA, we use a relatively small number of search agents, setting the escape possibility at 0.25, the population of wolves to be 5 and the visual distance at 1. The tolerance factor for the condition of stopping is equal to $\pm 1.0e^{-8}$. The results in terms of fitness values are averaged over 100 times for each optimization test function, by each length of the tabu list (u) which represents the strength of the memory of the wolf. WSA without any memory of previous visits is WSA [$u = 0$] or just WSA, and eWSA with full photographic memory of every step is eWSA [$u = \infty$] whose tabu list records the full amount of undesirable visited trails accumulated from all the previous generations. The computing environment is MATLAB running on a MacBook Pro (CPU speed: 2.3 gigahertz, RAM size: 4 gigabytes). The results of WSA and eWSA with varying tabu tenure from 5 to ∞ against the seven testing functions are shown in Table 1. The best results of each test function are formatted in boldface.

The second part of the experiment is the vis-a-vis comparison between eWSA and ACO where they both utilize some form of external memory for retaining heuristics. For the sake of a good balance between intensification and diversification (Yang, Deb & Fong, 2014; Yang, Deb, Hanne, & He, 2015), the parameters of the algorithms have been optimized beforehand; the initial default values are borrowed from the previous literature (Stützle & Hoos, 2000). The eWSA in this case is the eWSA with full record of visited places [$u = \infty$]. In the evaluation, both algorithms are subject to the seven testing functions which are presented in the Appendix; the fitness results are averaged for each of the seven functions. The experiment is repeated 100 times to avoid a possible bias resulting from the random effects of a single optimization run. At the start, the search agents are scattered randomly over the search space. The results are approximate solutions which may slightly differ from time to time. For the comparison, identical numbers of ants and wolves were used, $W = 20$. Both algorithms are executed exactly 500 iterations. The average fitness values resulting from 100 experiments of eWSA and ACO respectively are plotted as decreasing curves in Fig. 4 for the seven test problems under consideration. In five of the seven cases, eWSA obtains better results for a higher number of iterations while in the other two cases the results appear to be similar with a slight advantage of ACO.

The aspect of robustness of optimization runs has been considered by analyzing the standard deviations of best solutions obtained during the 100 repeated experiments for each method and each test problems. Table 2 shows the respective results for the generations 100, 200, 300, 400, and 500 with best values formatted in boldface. As can be seen in most cases eWSA leads to smaller standard deviations than ACO for the considered test problems.

In a third experiment series we consider higher dimensional problems (dimension d) which are solved by eWSA, PSO, and ACO. Just like in the second experiment, eWSA is applied with a full record of visited places ($u = \infty$), which has shown the best performance in the previous experiments when compared with other settings. The other parameters are the same as for the second experiment, i.e. there are 20 ants, particles and wolves assumed for the respective algorithms. We also use the same seven test functions as shown in the Appendix to do the evaluation. This time, however, we want to study the performance for higher dimensional problems and therefore compared with the previous experiments the dimension of each function is ten times higher than before. Each of the experiments is repeated 100 times.

Table 3 shows the results from the experiments in form of average values for each of the methods and each of the test problems. For four of the seven test functions, eWSA obtains the best results while PSO wins for three test functions. ACO never obtains the best results.

Table 1
Fitness values achieved by WSA and eWSA of varying memory terms.

eWSA version	Schaffer's	Sphere ($d = 10$)	Rosenbrock ($d = 3$)	Griewank ($d = 3$)	Michaelewicz ($d = 5$)	Rastrigin ($d = 2$)	Moved axis ($d = 2$)
WSA	0.0019	0.0029	0.0062	0.0062	−3.6215	0.2993	1.9606−06
eWSA [$u = 5$]	0.0026	0.0029	0.0085	0.0069	−3.5320	0.2157	6.2872E−07
eWSA [$u = 10$]	0.0032	0.0034	0.0053	0.0064	−3.4633	0.3009	3.9915E−07
eWSA [$u = 15$]	0.0022	0.0020	0.0090	0.0091	−3.6338	0.1994	6.5666E−07
eWSA [$u = 20$]	0.0032	0.0031	0.0049	**0.0042**	**−3.7373**	0.1117	1.1458E−07
eWSA [$u = 35$]	0.0032	0.0022	0.0156	0.0044	−3.3138	0.2017	4.4578E−07
eWSA [$u = 50$]	0.0029	0.0034	0.0080	0.0081	−3.5240	0.2003	**9.8157E−07**
eWSA [$u = 100$]	0.0013	0.0030	0.0142	0.0067	−3.5340	0.1016	1.3895E−07
eWSA [$u = 200$]	0.0019	0.0034	0.0111	0.0109	−3.4849	0.2020	1.8269E−07
eWSA [$u = 350$]	0.0019	0.0018	0.0128	0.0089	−3.5184	0.3062	9.0511E−07
eWSA [$u = 500$]	0.0016	0.0017	0.0065	0.0069	−3.4651	0.2110	3.110E−07
eWSA [$u = $ ALL]	**9.6651E−04**	**7.0102E−04**	**0.0022**	0.0067	−3.4542	**0.0062**	2.4962E−06

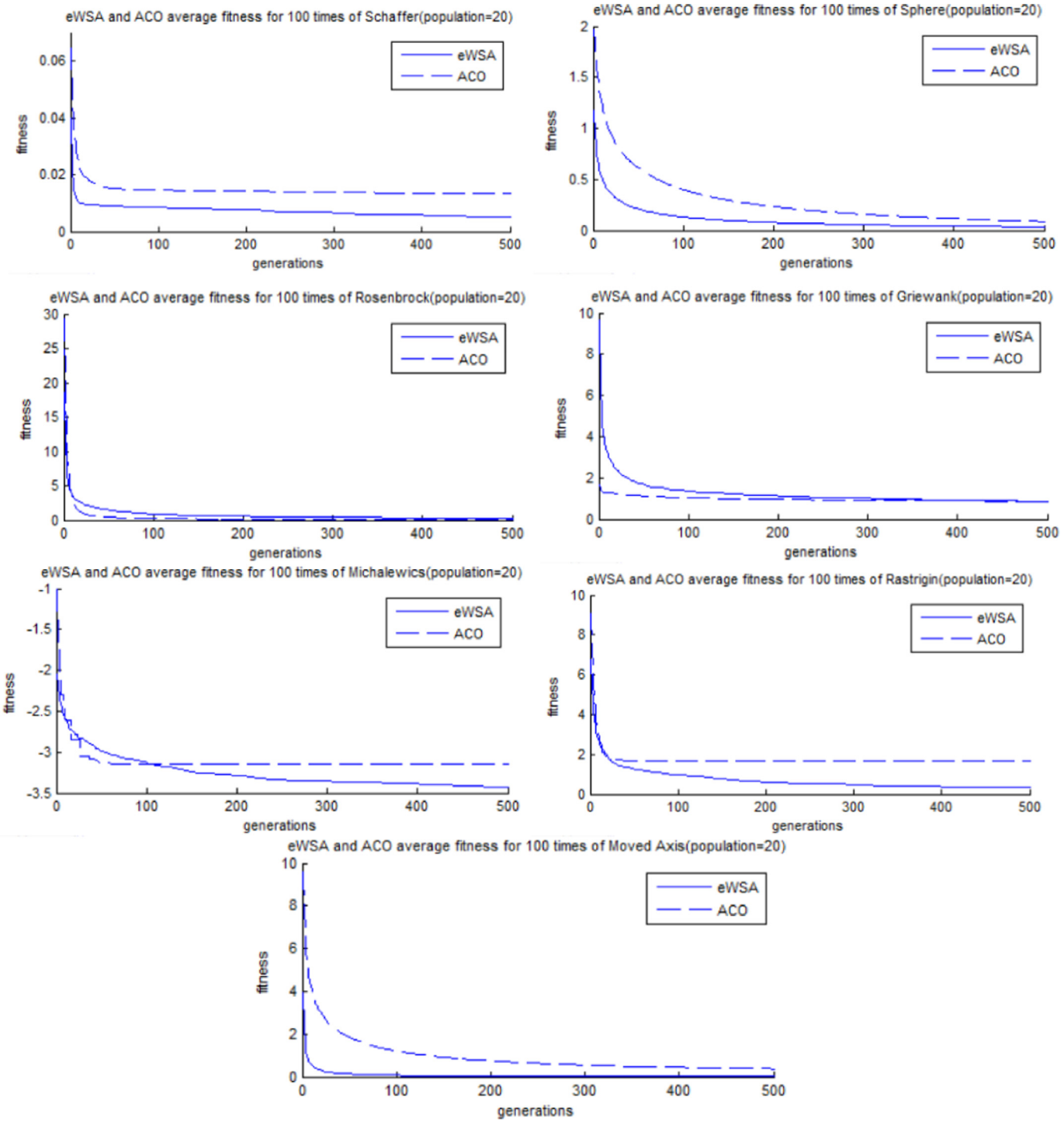

Fig. 4. Curve of fitness values obtained by eWSA and ACO for the considered seven test problems.

The last two columns of Table 3 show the *t*-test results for the comparison of results from ACO and eWSA and from PSO and eWSA (a comparison PSO and ACO is left out as this is not in the focus of our study). The *t*-test checks for a hypothesized mean difference between the considered methods' results of 0. We apply a two-sample *t*-test assuming unequal variances. The table shows the probability of the above null hypothesis. Except for Schaffer's F6 function this hypothesis must be rejected under all reasonable settings of alpha (error probability of rejecting the null hypothesis), e.g. alpha = 0.05. The differences in performance between the methods are, therefore, statistically significant. For Schaffer's F6 function all methods obtain almost the same results. eWSA is here slightly better but without statistical significance.

For analyzing speed and robustness of eWSA depending on its parameters values, we did a systematic analysis of different parameters for the number of wolves W, the visual range v, the velocity factor α, and the probability that a wolf does not leave the local proximity when any threat surfaces p_a. For W the values 2, 5, 10, 20, and 30 were considered. v was varied in steps of 0.1 in the interval [0.5, 1.5]. α was varied in steps of 0.1 in the interval [0.1, 0.9], and p_a was tested for the values 0.1, 0.2, 0.25, 0.3, and 0.4. All combinations of these parameter values were considered leading

Table 2
Standard deviations of best solutions for eWSA and ACO for the 7 test problems from 100 repetitions.

Generation	Method	Schaffer's F6	Sphere	Rosen	Griewank	Michalewicz	Rastrigin	Moved axis
100	eWSA	0.058	**0.181**	**0.509**	**0.246**	**0.244**	**0.488**	**0.059**
	ACO	**0.014**	0.399	0.599	8.915	0.432	1.349	1.921
200	eWSA	0.029	**0.174**	**0.306**	**0.220**	**0.253**	**0.419**	**0.030**
	ACO	**0.012**	0.234	0.425	8.911	0.421	1.349	1.352
300	eWSA	0.021	**0.140**	**0.292**	**0.243**	**0.230**	**0.356**	**0.020**
	ACO	**0.013**	0.158	0.362	8.909	0.421	1.349	1.068
400	eWSA	0.017	0.119	**0.272**	**0.226**	**0.207**	**0.302**	**0.018**
	ACO	**0.013**	**0.115**	0.332	8.907	0.415	1.349	0.899
500	eWSA	**0.013**	0.116	**0.274**	**0.223**	**0.247**	**0.286**	**0.017**
	ACO	**0.013**	**0.086**	0.290	8.906	0.417	1.349	0.766

Table 3
Experimental results for ACO, PSO and eWSA for higher dimensional test problems.

Test function	Dimension	Average fitness from 100 repetitions			*t*-test results	
		ACO	PSO	eWSA	eWSA-ACO	eWSA-PSO
Schaffer's F6	$d=20$	5.0000E−01	5.0000E−01	**4.9999E−01**	0.0596	0.2269
Sphere	$d=100$	6.5590E+02	**4.7085E+01**	5.6453E+02	1.63E−52	1.12E−164
Rosenbrock	$d=30$	1.1339E+03	1.0343E+03	**6.8396E+01**	7.60E−76	5.59E−52
Griewank	$d=30$	6.2621E+02	**3.0130E+01**	3.5297E+02	1.58E−73	1.86E−116
Michalewicz	$d=50$	−1.3484E+01	−1.9365E+01	**−2.1803E+01**	2.53E−95	3.11E−19
Rastrigin	$d=20$	1.9009E+02	1.1940E−01	**1.6815E+00**	6.68E−125	1.87E−17
Moved axis	$d=20$	3.6866E+03	**1.4987E+02**	2.1625E+03	1.11E−39	1.27E−90

Table 4
Experimental results for ACO, PSO and eWSA for low dimensional test problems with equal run time.

Test function	Dimension	Run time (seconds)	Number of iterations			Average fitness from 10 repetitions		
			ACO	PSO	eWSA	ACO	PSO	eWSA
Schaffer's F6	$d=2$	2	3002	2840	2557	*1.24E−01*	**2.21E−42**	1.12E−02
		3	4550	4305	3356	*7.20E−02*	**3.56E−69**	9.60E−03
		5	7489	7227	6098	*9.60E−02*	**1.94E−107**	8.46E−04
Sphere	$d=10$	2	2472	4322	2418	*9.19E+00*	**1.75E+00**	8.49E+00
		3	3731	6504	3527	7.16E+00	**1.77E+00**	*7.28E+00*
		5	6166	10,812	6005	*6.18E+00*	**1.73E+00**	3.51E+00
Rosenbrock	$d=3$	2	3695	4389	2493	*8.38E−02*	7.75E−02	**4.39E−02**
		3	5486	6500	3723	2.74E−04	**0.00E+00**	*2.84E−02*
		5	8756	10,796	6281	7.14E−05	**0.00E+00**	*6.74E−03*
Griewank	$d=3$	2	2670	2615	2112	*3.83E+00*	**1.29E−31**	6.65E−01
		3	4019	3980	3069	*2.20E+00*	**2.12E−52**	4.88E−01
		5	6556	6784	5231	*2.06E−01*	**5.71E−80**	1.13E−01
Michalewicz	$d=5$	2	2303	2320	2314	*−3.73E+00*	−3.74E+00	**−3.81E+00**
		3	3455	3516	3013	−4.48E+00	**−4.50E+00**	*−4.16E+00*
		5	5744	6087	4987	0.00E+00	−4.65E+00	**−4.68E+00**
Rastrigin	$d=2$	2	3809	4276	2202	*2.47E−01*	**0.00E+00**	7.13E−02
		3	5671	6411	3689	*1.99E−01*	**0.00E+00**	7.36E−02
		5	9113	11,014	6216	*9.95E−02*	**0.00E+00**	2.04E−02
Moved axis	$d=2$	2	4033	4905	2484	7.32E−10	**0.00E+00**	*4.84E−03*
		3	6006	7365	3710	5.11E−11	**0.00E+00**	*1.61E−03*
		5	9749	12,203	6304	1.51E−11	**0.00E+00**	*1.04E−04*

to 2475 test runs. These test runs were conducted with the seven test functions discussed above.

Interestingly, there is no obvious result concerning parameter values which are in general favorable, i.e. each test function has best results for parameter values combinations which can be quite different. There is some indication that the population size and visual distance should not be too small, i.e. W should be between 10 and 20 (rather 10 than 20) and v between 1 and 1.4, but good values for the other two parameters seem to be rather unstable (with respect to the chosen test function). Considering rather the robustness of parameter values than their average performance, we have chosen the following setting $W=10$, $v=1.2$, $\alpha=0.2$, and $p_a=0.3$.

Based on these parameter values, a fourth series of experiments was conducted in order to compare eWSA, PSO, and ACO again for lower dimensional problems (dimension d varying for the considered test function between 2 and 10 as in experiment series 1 and 2) and under a setting with dedicating equal run time to each algorithm being executed on the same computer. As before, eWSA is applied with a full record of visited places ($u=\infty$). PSO and ACO were applied with the parameter settings mentioned above. For all test settings 10 repetitions were performed.

The results in Table 4 show the number of performed iterations (generations) for each algorithm and each test problem for test runs of 2 seconds, 3 seconds, and 5 seconds. Accordingly for each test scenario, the obtained fitness values are shown as well. The best results from each test scenario are printed in boldface while the worst are shown in italic. It is obvious that eWSA is slower in terms of number of iterations than the other two algorithms: during the same time, PSO can perform almost 60 percent more iterations than eWSA and ACO still more than 30 percent more iterations. Nevertheless, in a majority of cases, eWSA outperforms ACO while PSO mostly outperforms eWSA (eWSA is the best method in 3 cases).

During these tests we also analyzed the utilization of the tabu list for eWSA: For each setting, the number of hits (finding relevant tabu list entries) lies between 9.71 percent and 17.63 percent per generation, or respectively between 0.97 percent and 1.76 percent per wolf. Note that the last values are independent from the number of wolves whereas the first values increase with a larger population size. The observed hit rate appears to be smaller for easier test function (like the Moved axis function) and higher for more difficult and multimodal cases (like the Rastrigin function).

5. Discussion

In the low-dimensional experiment that tested the effects of eWSA versus standard WSA, the winning configuration is the eWSA with full memory tenure that wins four out of seven testing algorithms. The fitness convergence is improved notably when the full trails are memorized in GMS and prohibited from revisiting during the search. eWSA with $u = 20$ and $u = 50$ won three times altogether within the group of three algorithms. Incidentally that shows the memory tenure for eWSA requires certain length in order to take effect. In contrast, when eWSA is equipped with short-term memory (e.g. $u = 5$), the fitness is not better than standard WSA. Overall, the experiment shows that eWSA indeed, improves the quality of the found solution in terms of better fitness. However, the tabu list must be of at least, certain length for enabling the performance advantage. If the search space is very large, it requires the search agents to save up a considerable large amount of history trails so as to realize the improvement.

As shown in Fig. 4, although eWSA yields in most cases a relatively better average fitness than ACO, the fitness curve for eWSA takes often longer to settle into a steady-state than ACO. For most of the considered test problems (in 5 from 7 cases) it can be seen that even at the end of the observed iterations (i.e. between iterations 400 and 500) still further improvements occur while ACO shows in most cases (at least in 4 from 7 cases) no visible further improvements after 400 iterations. This phenomenon may be due to the nature of the metaheuristic algorithms – eWSA is known for its semi-swarm characteristic where the agents go diversified in their own ways and only occasionally they merge. The wolves that search somewhat separately and make far and wide movements facilitate discovering a better solution. The advantage however only gets revealed in the later generation after some intensive searching around. This may explain similarly the advantage of the eidetic memory of the wolves emerge in a later stage only after they accumulate a significantly long history of past visits. The early convergence by ACO may be factored by the flocking behaviors of the ants that collectively and progressively construct the pheromone paths leading steadily to the final solution.

Given an equal number of search agents, our experiments demonstrate that eWSA mostly outperforms ACO, as shown in Fig. 4. Interestingly, ACO occasionally beats eWSA in fitness performance in the middle period of generations, e.g. during iteration 30 and 100 for the Michalewicz's test function. Again this attests the phenomenon that eWSA spreads out the search agents in the early stage, thinning out the chances of excavating good quality solutions. But given a sufficiently long time, over many generations, the widespread search may start to yield better results especially when the eidetic memory reserve matures. The difference (gain) of fitness' between eWSA over ACO increases usually in the late stage of the optimization runs. The increase in the difference of performance slows down as both algorithms appear to converge into a global optimal.

The results of the third experiment show that eWSA also performs quite well for higher dimensional problems of the considered types. In comparison with ACO and PSO, eWSA wins for four of the seven test functions. PSO wins in three test cases while ACO is always the worst performing method. For six of the seven test functions, the results can be assumed as statistically significant based on the obtained t-test results. It is interesting to note that all the methods have obviously more difficulties to come close to the optimum in the higher dimensional situations. Especially, for the Rosenbrock and the Moved axis functions, the results are comparably poor. As the diverse results of the test functions affected by high multimodality (Rastrigin's function, Griewangk's function, and Michalewicz's function) show, multimodality does not seem to have the highest impact on the methods' performance and in particular, eWSA seems to cope well with that. However, even for the simple sphere function there is still clear potential for further improvements of the methods' results.

The results of the fourth experiment are certainly less compelling concerning the eWSA performance. It is not surprising that a single iteration of the algorithm requires more time than, for instance, a PSO iteration as PSO is more simple concerning the algorithmic logic. Unfortunately, in most cases, this disadvantage cannot be compensated by a faster convergence, i.e. reaching a particular fitness in fewer generations. The faster convergence exists as the results from the other experiments confirm but cannot compensate the ca. 60 percent longer runtime per iteration (compared with PSO). eWSA only wins in 3 from 21 cases. In comparison with ACO the run time disadvantage is smaller, about 30 percent per iteration and the faster convergence of eWSA can mostly compensate that (better than ACO in 14 of 21 cases).

We have analyzed the time consumption of eWSA in more details by profiling the MATLAB code. As most time consuming, the calculation of distances of wolves was detected. By replacing the standard MATLAB functions and taking into account the symmetry of the distance function small time savings could be obtained. The second most time consuming step is the generation of random numbers while the third most time consuming step is the calculation of fitness values, i.e. the evaluation of test functions. The time spent for random number generation could also possibly be improved by not using the standard MATLAB functions and, instead, employing a faster random number generator (see, e.g. L'Ecuyer & Panneton, 2005). In order to speed-up the algorithm, another direction for future work could be the shift to parallel architectures and to provide a wolf search algorithm with derives advantage of them.

6. Conclusion

For hard-to-solve optimization problems, approximate solutions discovered by stochastic search algorithms are often superior to more traditional optimization approaches in terms of running time. Recently in the optimization research field, swarm intelligence algorithms demonstrated their effectiveness as good problem solvers across wide application domains. Especially, Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) have become popular choices of metaheuristics because they were found successful for solving different types of optimization problems. We are inspired by ACO, where each ant searches for the solution according to the memory of the pheromone trails which are accumulated from the solutions of the past iterations. In this paper, we have proposed a memory-based wolf search algorithm called Eidetic WSA for efficiently remembering the visited places in the search space by the wolves. In the implementation, a tabu list programmed using a linked list is used as a memory of variable length that stores the visited places in terms of solutions found previously, and can use them to steer the direction of search for constructing a new solution. The proposed algorithm has been evaluated rigorously by seven classical optimization testing functions in a series of computer simulations. The results clearly demonstrated that the proposed algorithm works better (for the

same number of iterations) than the standard WSA when the tabu list is of sufficient length. The new algorithm is compared with ACO and PSO for seven test problems and superior results are obtained after the new algorithm has reached a late generation where eWSA, however, shows longer run times for the same number of iterations. In terms of computation time, the new algorithm can beat only ACO as its single iterations are more time consuming than those of PSO.

For future research, more comprehensive tests are suggested including the comparison of Eidetic WSA with a broader range of other optimization methods and considering other types of optimization problems, in particular, combinatorial optimization problems. For such kind of problems the usage of the tabu list may become even more important due to a more frequent reoccurring of solutions and higher hit rates. Moreover, there is certainly also potential to further speed-up the algorithm, e.g. by using improved data structures for the tabu list. Future experiments will also include comparisons with further metaheuristics such as the improved ACO (ACO_R) by Socha and Dorigo (2008) which should show better results for continuous optimization problems than the usual ACO.

Acknowledgment

The authors are thankful for the financial support from the research grant "Temporal Data Stream Mining by Using Incrementally Optimized Very Fast Decision Forest (iOVFDF)," Grant no. MYRG2015-00128-FST, offered by the University of Macau, FST, and RDAO.

Appendix. Fitness functions

We used the following test functions with dimensionality d to test our algorithms.

1. Griewangk's function: in this test function, there are a lot of local minima,

$$f(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

The global minimum is $f_{min} = 0$ at (0,...,0), and x is bounded by $-600 \le x_i \le 600$.

2. Sphere function:

$$f(x) = \sum_{i=1}^{d} x_i^2, \quad -5.12 < x_i < 5.12$$

The global minimum is $f_{min} = 0$ at (0,...,0).

3. Rastrigin's function: this function is difficult due to its large search space and large number of local minima.

$$f(x) = Ad + \sum_{i=1}^{d} \left[x_i^2 - A\cos(2\pi x_i)\right],$$

where $A = 10$, $x_i \in [-5.12, 5.12]$ and the global minimum is $f_{min} = 0$ at (0,...,0).

4. Schaffer F6:

$$f(x) = 0.5 + \frac{\sin^2\sqrt{\sum_{i=1}^{d} x_i^2} - 0.5}{\left(1 + 0.001 \times \sum_{i=1}^{d} x_i^2\right)^2}$$

where $x_i \in [10, 10]$ and the global minimum is $f_{min} = 0$ at (0,...,0).

5. Moved axis parallel hyper-ellipsoid function:

$$f(x) = \sum_{i=1}^{d} 5i \cdot x_i^2$$

where $x_i \in [-5.12, 5.12]$and the global minimum is $f_{min} = 0$ at (0,...,0).

6. Michalewicz's function:

$$f(x) = -\sum_{i=1}^{d} \sin(x_i)\left[\sin\left(\frac{ix_i^2}{\pi}\right)\right]^{2m}$$

where $m = 10$ and $x_i \in [0, \pi]$. When $d = 2$, the global minimum is $f_{min} = -1.803$ at position (2.0230, 2.0230).

7. Rosenbrock function:

$$f(x) = \sum_{i=1}^{d-1} [(1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2]$$

where $x_i \in [-100, 100]$. The global minimum is $f_{min} = 0$ at position (1,...,1).

References

Acan, A. (2004). An external memory implementation in ant colony optimization. In *Proceedings of international workshop on ant colony optimization and swarm intelligence (ANTS2004)* (pp. 73–82).

Acan, A. (2005). An external partial permutation memory for ant colony optimization. In *Proceedings of EvoCOP 2005, Volume 3448 of Lecture notes in computer science* (pp. 1–11). Berlin, Heidelberg: Springer.

Charansiriphaisan, K., Chiewchanwattana, S., & Sunat, K. (2013). A comparative study of improved artificial bee colony algorithms applied to multilevel image thresholding. *Mathematical Problems in Engineering, 2013*, 17. doi:10.1155/2013/927591.

Cheng, X., & Lin, Y. (2014). Multiobjective robust design of the double wishbone suspension system based on particle swarm optimization. *The Scientific World Journal, 2014*, 7. doi:10.1155/2014/354857.

Chiang, T.-A., Che, Z. H., & Cui, Z. (2014). Designing a multistage supply chain in cross-stage reverse logistics environments: Application of particle swarm optimization algorithms. *The Scientific World Journal, 2014*, 19. doi:10.1155/2014/595902.

Dorigo, M. (2001). Ant algorithms solve difficult optimization problems. *Advances in artificial life* (pp. 11–22). Berlin, Heidelberg: Springer.

Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science: Vol. 1* (pp. 39–43).

Eggermont, J., & Lenaerts, T. (2001). *Non-stationary function optimization using evolutionary algorithms with a case-based memory. Technical Report 2001-11*. Leiden University Advanced Computer Science (LIACS).

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research, 13*(5), 533–549.

Glover, F. (1989). Tabu search – Part 1. *ORSA Journal on Computing, 1*(2), 190–206.

Goldberg, D. (1989). *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley.

Goñi, A., & Cordero, P. (2009). Global memory structure for ant colony optimization algorithms. In Intelligent Engineering, *International book series on information science and computing*, Book 11: Intelligent Engineering, Sofia: ITHEA (pp. 116–122).

Guntsch, M., & Middendorf, M. (2002). A population based approach for ACO. In Proceedings of applications of evolutionary computing – EvoWorkshops2002, *Volume 2279 of Lecture notes in computer science* (pp. 72–81). Berlin, Heidelberg: Springer.

Hani, Y., Amodeo, L., Yalaoui, F., & Chen, H. (2007). Ant colony optimization for solving an industrial layout problem. *European Journal of Operational Research, 183*(2), 633–642.

L'Ecuyer, P., & Panneton, F. (2005). Fast random number generators based on linear recurrences modulo 2: Overview and comparison. In *Proceedings of the winter simulation conference* (p. 10). IEEE.

Liao, T., Stützle, T., de Oca, M. A. M., & Dorigo, M. (2014). A unified ant colony optimization algorithm for continuous optimization. *European Journal of Operational Research, 234*(3), 597–609.

Lin, M.-H., Tsai, J.-F., & Lee, L.-Y. (2013). Ant colony optimization for social utility maximization in a multiuser communication system. *Mathematical Problems in Engineering, 2013*, 8. Article ID 798631. doi:10.1155/2013/798631.

Ma, L., Chen, H., Hu, K., & Zhu, Y. (2014). Hierarchical artificial bee colony algorithm for RFID network planning optimization. *The Scientific World Journal, 2014*, 21. doi:10.1155/2014/941532.

Mavrovouniotis, M., & Yang, S. (2011). Memory-based immigrants for ant colony optimization in changing environments. *Applications of evolutionary computation, Volume 6624 of Lecture notes in computer science* (pp. 324–333). Berlin, Heidelberg: Springer.

Montgomery, J., & Randall, M. (2003). The accumulated experience ant colony for the travelling salesman problem. *International Journal of Computational Intelligence and Applications, 3*(2), 189–198.

Nada, M., & Al-Salami, A. (2009). Ant colony optimization algorithm. *UbiCC Journal, 4*(3), 823–826.

Pan, Q. K. (2016). An effective co-evolutionary artificial bee colony algorithm for steelmaking-continuous casting scheduling. *European Journal of Operational Research, 250*(3), 702–714.

Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization. *Swarm Intelligence, 1*(1), 33–57.

Rupérez Cañas, D., Sandoval Orozco, A. L., García Villalba, L. J., & Hong, P.-S. (2013). Hybrid ACO routing protocol for mobile ad hoc networks. *International Journal of Distributed Sensor Networks, 2013*, 7. doi:10.1155/2013/265485.

Salto, C., Leguizamón, G., & Alba, E. (2009). External memory in a hybrid ant colony system for a 2D strip packing. In *Proceedings of the XV Congreso Argentino de Ciencias de la Computación, October 2009* (pp. 30–39). http://hdl.handle.net/10915/20880.

Schyns, M. (2015). An ant colony system for responsive dynamic vehicle routing. *European Journal of Operational Research, 245*(3), 704–718.

Sermpinis, G., Theofilatos, K., Karathanasopoulos, A., Georgopoulos, E. F., & Dunis, C. (2013). Forecasting foreign exchange rates with adaptive neural networks using radial-basis functions and particle swarm optimization. *European Journal of Operational Research, 225*(3), 528–540.

Socha, K., & Dorigo, M. (2008). Ant colony optimization for continuous domains. *European Journal of Operational Research, 185*(3), 1155–1173.

Sreeja, N. K., & Sankar, A. (2015). Ant colony optimization based binary search for efficient point pattern matching in images. *European Journal of Operational Research, 246*(1), 154–169.

Stützle, T., & Hoos, H. (2000). Max-min ant system. *Future Generation Computer Systems, 16*(8), 889–914.

Tang, R., Fong, S., Yang, X.-S., & Deb, S. (2012). Wolf search algorithm with ephemeral memory. In *Proceedings of the IEEE Seventh international conference on digital information management (ICDIM 2012), August 22–24, 2012* (pp. 165–172).

Törn, A., & Zilinskas, A. (1989). Global optimization. *Lecture notes in computer science*. Berlin, Heidelberg: Springer.

Tseng, C. T., & Liao, C. J. (2008). A discrete particle swarm optimization for lot-streaming flowshop scheduling problem. *European Journal of Operational Research, 191*(2), 360–373.

Tsutsui, S. (2006). cAS: Ant colony optimization with cunning ants. In Proceedings of the IX conference on parallel problem solving from nature – PPSN IX, *Volume 4193 of Lecture notes in computer science* (pp. 162–171). Berlin, Heidelberg: Springer.

Wang, G. G., Deb, S., & Cui, Z. (2015). Monarch butterfly optimization. *Neural Computing and Applications*, 1–20, in print.

Wang, G. G., Gandomi, A. H., Alavi, A. H., & Deb, S. (2015). A hybrid method based on krill herd and quantum-behaved particle swarm optimization. *Neural Computing and Applications*, 1–18, in print.

Wang, R.-L., & Zhao, L.-Q. (2010). A memory-based ant colony algorithm for the bipartite subgraph problem. *IJCSNS International Journal of Computer Science and Network Security, 10*(3), 235–240.

Wang, R.-L., Zhao, L.-Q., & Zhou, X.-F. (2012). Ant colony optimization with memory and its application to traveling salesman problem. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E95.A*(3), 639–645.

Xiao, Y., Xia, L., Zhu, S., Huang, D., & Xie, J. (2013). Video shot boundary recognition based on adaptive locality preserving projections. *Mathematical Problems in Engineering, 2013*, 9. doi:10.1155/2013/353261.

Yang, X.-S., & Deb, S. (2013). Multiobjective cuckoo search for design optimization. *Computers & Operations Research, 40*(6), 1616–1624.

Yang, X.-S., Deb, S., & Fong, S. (2014). Metaheuristic algorithms: Optimal balance of intensification and diversification. *Applied Mathematics & Information Sciences, 8*(3), 977–983.

Yang, X.-S., Deb, S., & Fong, S. (2014). Bat algorithm is better than intermittent search strategy. *Journal of Multi-Valued Logic & Soft Computing, 22*(3), 223–237.

Yang, X.-S., Deb, S., Hanne, T., & He, X. (2015). Attraction and diffusion in nature-inspired optimization algorithms. *Neural Computing and Applications*, 1–8, in print.

Yin, P. Y., Glover, F., Laguna, M., & Zhu, J. X. (2010). Cyber swarm algorithms – Improving particle swarm optimization using adaptive memory strategies. *European Journal of Operational Research, 201*(2), 377–389.

Zhang, L., & Wu, J. (2014). A PSO-based hybrid metaheuristic for permutation flowshop scheduling problems. *The Scientific World Journal, 2014*, 8. doi:10.1155/2014/902950.