

# An Enhanced Aggregation Pheromone System for Real-Parameter Optimization in the ACO Metaphor

Shigeyoshi Tsutsui

Hannan University, Matsubara, Osaka 580-8502, Japan  
tsutsui@hannan-u.ac.jp

**Abstract.** In previous papers we proposed an algorithm for real parameter optimization called the Aggregation Pheromone System (APS). The APS replaces pheromone trails in traditional ACO with aggregation pheromones. The pheromone update rule is applied in a way similar to that of ACO. In this paper, we proposed an enhanced APS (eAPS), which uses a colony model with *units*. It allows a stronger exploitation of better solutions found and at the same time it can prevent premature stagnation of the search. Experimental results showed eAPS has higher performance than APS. It has also shown that the parameter settings for eAPS are more robust than for APS.

## 1 Introduction

Ant colony optimization (ACO) has been applied with great success to a large number of discrete optimization problems [1–7]. However, up to now, few ant-colony based approaches for continuous optimization have been proposed in the literature. The first such method, called Continuous ACO (CACO), was proposed in [8]. CACO combines ACO with a real-coded GA, with the ACO approach being largely devoted to local search. CACO was extended in [9, 10]. In [11], the behavior of an ant called *Pachycondyla Apicalis* was introduced as the base metaphor of the algorithm (API) to solve continuous problems. The Continuous Interacting Ant Colony (CIAC) in [12] also introduces an additional direct communication scheme among ants.

In contrast to the above studies, studies that have a purely pheromone based method were proposed independently in [13, 14]. In [13], the pheromone intensity is represented by a single normal probability distribution function. The center of the normal distribution is updated at each iteration to the position which has the best functional value. The variance value is updated according to the current ant distribution. In [14], a mixture of normal distributions was introduced to solve multimodal functions. However, both of the above two approaches use marginal distribution models and the pheromone update rules used are quite different than those of the original ACO methods.

In [15, 16], we proposed a model called the Aggregation Pheromone System (APS). The APS replaces pheromone trails in traditional ACO with aggregation pheromones and uses them as the base metaphor of the algorithm. The

pheromone update rule is applied in a way similar to that of AS [1], and as a result, the aggregation pheromone density eventually becomes a mixture of multivariate normal distributions. APS could solve even hard problems which have a tight linkage among parameters. However, this model was sensitive to control parameters.

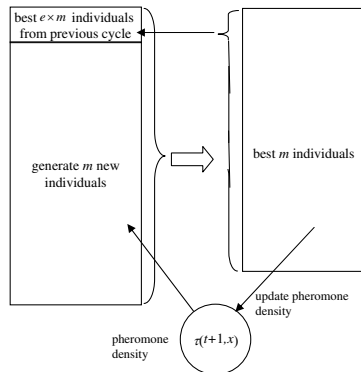
In this paper, we propose an enhanced APS (eAPS) which is more robust to the control parameters and has higher performance than APS. This is achieved by applying stronger exploitation of better solutions found during the search and by using a colony model with *units*. The remainder of this paper is organized as follows. Section 2 gives a brief review of APS. Section 3 describes how the eAPS is structured. Section 4 presents experimental results. Finally, Section 5 concludes the paper.

## 2 A Review of Aggregation Pheromone System (APS)

In the real world, *aggregation pheromones* are used by some insect species to communicate with members of their community to share information about the location of food, safe shelter, potential mates, or enemies. Many types of function of aggregation behavior have been observed. These include foraging-site marking, mating, finding shelter, and defense. When they have found safe shelter, cockroaches produce a specific pheromone in their excrement which attracts other members of their species [17].

The difference between ACO and APS is in how the pheromones function in the search space [15, 16]. In ACO, pheromone intensity is defined as a trail on a node or an edge between nodes of a given sequencing problem. On the other hand, in APS, the aggregation pheromone density is defined by a continuous density function in the search space  $X$  in  $R^n$ .

Fig.1 shows the colony model of the previously proposed APS. Here, the colony consists of a population of  $m$  individuals. The aggregation pheromone density  $\tau(t,x)$  is defined in the search space  $X$  in  $R^n$ . Initially ( $t=0$ ), the aggregation pheromone is distributed uniformly. The initial pheromone density is used to generate the first  $m$  individuals; therefore, the initial individuals are generated randomly with a uniform distribution over all possible individuals. Next, the individuals are evaluated and aggregation pheromone is emitted by  $m$  individuals depending on their functional values. Then the pheromone density for the next iteration  $\tau(t+1,x)$  is obtained and  $m$  individuals for next iteration are generated according to  $\tau(t+1,x)$ .



**Fig. 1.** Colony model of the APS

Here the best individuals of  $e \times m$  from the previous population are added to the newly generated solutions (the value of  $e$  should be rather small, e.g.  $e = 0.1$ ). The best  $m$  individuals are selected from the combined population of  $(e+1) \times m$  to create the next population of  $m$  individuals. Setting  $t \leftarrow t+1$ , this iteration continues until the predetermined termination criteria are satisfied. Note here that in APS, the best  $e \times m$  individuals are retained as candidate individuals for the next iteration so that APS can perform a stronger exploitation of best solutions found in the search.

### 3 Enhanced Aggregation Pheromone System (eAPS)

#### 3.1 The Model of the eAPS

In eAPS, we use a colony model as shown in Fig.2. The colony model consists of  $m$  units, and each unit consists of only one individual. At iteration  $t+1$ , a new individual  $x_{i,t+1}^{new}$  ( $i = 1, 2, \dots, m$ ) is generated according to the pheromone density  $\tau(t+1, x)$ . It is then compared with the existing individual  $x_{i,t}^*$  in the unit. If  $x_{i,t+1}^{new}$  is better than  $x_{i,t}^*$ , it is replaced with  $x_{i,t+1}^{new}$  and is indicated as  $x_{i,t+1}^*$  (here note that the notation of individual  $x_{i,t}^{new}$  (or  $x_{i,t}^*$ ) also represents the vector value of the individual).

Thus in eAPS, the best individual of each unit is always retained. After the best individual for each unit is obtained and iteration index is updated, aggregation pheromone is emitted by all  $x_{i,t}^*$  ( $i = 1, 2, \dots, m$ ) depending on their functional values. This colony model has following two important features:

1. It has a stronger exploitation of the best individuals than APS in Fig. 1 since each unit maintains the best individual found so far in the unit.
2. The comparison method in each unit is similar to tournament selection in genetic algorithms (GAs), but it differs from traditional tournament selection because the comparison is performed only between existing and newly generated individuals. However, just as tournament selection can maintain diversity of a population in GAs, this colony model can also maintain the diversity of  $x_{i,t}^*$ .

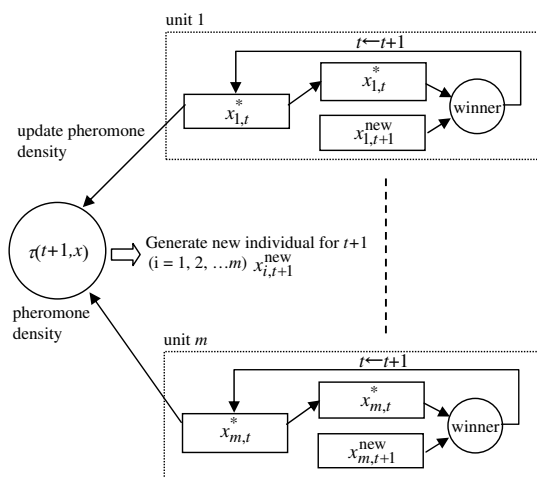


Fig. 2. Colony model of the eAPS

### 3.2 Updating the pheromone intensity and sampling new individuals

Although the methods of updating the pheromone intensity and sampling new individuals in eAPS are basically the same as with APS, here we describe them in a more schematic manner.

**3.2.1 Probability density function of pheromone** In each iteration,  $x_{i,t}^{new}$  ( $i=1,2, \dots, m$ ) individuals are generated based on the density of the aggregation pheromone. Individuals are attracted more strongly to positions where the pheromone density is higher, whereas they are attracted less strongly to positions where the density is lower. More specifically, eAPS generates new individuals using probabilities proportional to the values of the aggregation density function. Let  $\tau(t, x)$  be the density function of the aggregation pheromone in iteration  $t$  in search space  $X$ . Initially ( $t=0$ ), the aggregation pheromone is distributed uniformly, that is,  $\tau(0, x) = c$  where  $c$  is an arbitrary positive constant. The probability density function used to generate new candidate solutions at iteration  $t$ , denoted by  $p_\tau(t, x)$ , is defined as

$$p_\tau(t, x) = \frac{\tau(t, x)}{\int_X \tau(t, x) dx}. \quad (1)$$

Note that each individual  $x_{i,t}^*$  emits aggregation pheromone in its neighborhood. The intensity of aggregation pheromone emitted by individual  $x_{i,t}^*$  is based on the following properties:

1. The intensity of the pheromone emitted by  $x_{i,t}^*$  depend on its functional value  $f(x_{i,t}^*)$  – the higher the functional value of  $x_{i,t}^*$ , the higher the pheromone intensity emitted by  $x_{i,t}^*$ . We use ranking based on functional value to control pheromone intensity. The best individual has a rank  $m$ , whereas the worst individual has a rank 1.
2. The intensity emitted by  $x_{i,t}^*$  should decrease with distance so that the pheromone intensity of the points that are closer to  $x_{i,t}^*$  is increased more than the pheromone intensity of the points that are further from  $x_{i,t}^*$ .
3. In order to alleviate the effects of linear transformations of the search space, the intensity should consider the overall distribution of individuals in each unit. To achieve this, the pheromone intensity emitted by  $x_{i,t}^*$  was chosen to be a Gaussian distribution with covariance equal to the covariance of  $x_{i,t}^*$  ( $i=1, 2, \dots, m$ ).

Let us represent the rank value of individual  $x_{i,t}^*$  at iteration  $t$  by  $r(t, x_{i,t}^*)$ . Then, the density of the pheromone intensity emitted by  $x_{i,t}^*$  at point  $x$  is given by a scaled multivariate normal distribution as

$$\Delta\tau'(t, x_{i,t}^*, x) = C \frac{(r(t, x_{i,t}^*))^\alpha}{\sum_{j=1}^m j^\alpha} N(x; x_{i,t}^*, \beta^2 \Sigma_t), \quad (2)$$

where  $N(x; x_{i,t}^*, \beta^2 \Sigma_t)$  is a multivariate normal distribution function (its center is located at  $x_{i,t}^*$ ),  $\Sigma_t$  is the covariance matrix of all individuals of units at iteration  $t$ ,  $\alpha$  ( $\alpha > 0$ ) is a parameter for adjusting the relative importance of rank,  $\beta$  ( $\beta > 0$ ) is a scaling factor, and  $C$  is the total pheromone intensity emitted from all units at iteration  $t$ . The total aggregation pheromone density emitted by the  $m$  individuals  $x_{i,t}^*$  ( $i = 1, 2, \dots, m$ ) at iteration  $t$  is then given by

$$\Delta\tau(t, x) = \sum_{i=1}^m \Delta\tau'(t, x_{i,t}^*, x). \quad (3)$$

Note that  $C$  is assumed to be constant for all  $t$ , that is,

$$\int_X \Delta\tau(t, x) dx = C \text{ for } t \geq 0. \quad (4)$$

The sampling method in 3.2.2 is based on the assumptions of Eq. 4. The scaling factor  $\beta$  controls exploration by reducing the amount of variation as the colony converges to the optimum. The total aggregation pheromone density is updated according to the following formula as in ACO:

$$\tau(t+1, x) = \rho \cdot \tau(t, x) + \Delta\tau(t, x), \quad (5)$$

where  $\rho$  ( $0 \leq \rho < 1$ ) controls the evaporation rate.

After the pheromone updating is performed, a new individual  $x_{i,t+1}^{new}$  is generated based on the new pheromone density  $\tau(t+1, x)$ ,  $x_{i,t+1}^*$  is obtained by comparison between  $x_{i,t}^*$  and  $x_{i,t+1}^{new}$  at each unit  $i$ , and the next iteration of eAPS is performed. Since the pheromone density updates increase pheromone intensity near individuals with better functional value, the pheromone density in promising regions of the search space is expected to increase over successive iterations, eventually converging to global optima.

**3.2.2 Sampling new individuals** As described in 3.2.1, new individuals are sampled with probabilities proportional to the aggregation density  $\tau(t+1, x)$ . To perform the sampling, we need to obtain the probability density function  $p_\tau(t+1, x)$  from  $\tau(t+1, x)$ . Since  $\tau(t+1, x)$  in Eq. 5 can be rewritten as

$$\tau(t+1, x) = \rho^{t+1} \tau(0, x) + \sum_{h=0}^t \rho^h \Delta\tau(t-h, x). \quad (6)$$

$p_\tau(t+1, x)$  is obtained from Eqs. 1, 4, and 6 as

$$p_\tau(t+1, x) = \frac{\rho^{t+1}}{\sum_{k=0}^{t+1} \rho^k} \cdot \frac{\tau(0, x)}{C} + \sum_{h=0}^t \frac{\rho^h}{\sum_{k=0}^{t+1} \rho^k} \cdot \frac{\Delta\tau(t-h, x)}{C}. \quad (7)$$

Here note that we assume  $\rho^0 = 1$  for  $0 \leq \rho < 1$  for the sake of convenience.

In general, if a probability density function  $f(x)$  can be written as a mixture of other probability density functions

$$f(x) = p_1 f_1(x) + p_2 f_2(x) + \dots + p_S f_S(x) \quad (8)$$

with  $p_1 + p_2 + \dots + p_S = 1$ , then the sampling of each point according to the distribution defined by  $f(x)$  proceeds as follows:

1. Choose which mixture component  $f_s(x)$  should be used (a mixture component  $f_s(x)$  is chosen with probability  $p_s$ ).
2. Generate a random point according to the density function of the chosen component.

Since  $p_\tau(t+1, x)$  is indeed a mixture distribution of  $t+1$  multivariate Gaussian distributions and one uniform distribution, we can sample it using the standard sampling procedure for mixture distributions where the probability of the  $s$ -th component of the mixture is

$$p_s = \rho^s / \sum_{k=0}^{t+1} \rho^k \quad (9)$$

and the  $s$ -th mixture component is given by

$$f_s(x) = \begin{cases} \Delta\tau(t-s, x)/C & \text{if } s \leq t \\ \tau(0, x)/C & \text{otherwise.} \end{cases} \quad (10)$$

This sampling is called *iteration sampling*. Sampling according to the last mixture component  $f_{t+1}(x)$  is simple because  $f_{t+1}(x)$  is a constant and thus represents a uniform distribution over the entire search space. The remaining components,  $f_s(x)$ , where  $s \leq t$ , are mixture distributions of  $m$  components each:

$$\frac{\Delta\tau(t-s, x)}{C} = \sum_{i=1}^m \frac{(r(t-s, x_{i,t-s}^*))^\alpha}{\sum_{j=1}^m j^\alpha} N(x; x_{i,t-s}^*, \beta^2 \Sigma_{t-s}). \quad (11)$$

Sampling according to  $f_s(x)$ , where  $s \leq t$ , can be done in a similar way, i.e., by using the sampling procedure for mixture distributions where the probability of the  $i$ -th component  $N(x; x_{i,t-s}^*, \beta^2 \Sigma_{t-s})$  of Eq. 11 is

$$p_i = (r(t-s, x_{i,t-s}^*))^\alpha / \sum_{j=1}^m j^\alpha. \quad (12)$$

We call this sampling *rank sampling*. Since each component is a normal distribution, it can be sampled using Cholesky decomposition [18].

To perform the sampling based on Eq. 7, using the above sampling method, we need a large amount of memory to store  $x_{r,t-h}^*$  vector values and the covariance matrix  $\beta^2 \Sigma_{t-h}$  when the iteration index  $t$  becomes large. In this case  $\rho^h \rightarrow 0$  for large  $h$  since  $\rho < 1$ . Thus, we can limit the maximum number of iterations to keep data up to a constant  $H$ . For  $t \geq H$  we need to use slightly modified probability function  $p_\tau(t+1, x)$  from Eq. 7.

Although at each iteration sampling is performed probabilistically according to Eq. 7 (or its modification for  $t \geq H$ ), we also introduce a perturbation, resulting from conflict among individuals, or environmental disturbances. Perturbation works to perform the same function as mutation in evolutionary algorithms. The perturbation rate is represented by  $P_{rate}$ . The pseudo-code of eAPS is shown in Fig. 3.

1.  $t \leftarrow 0$
2. Set the initial pheromone density  $\tau(0, x)$  uniformly
3. Sample two individuals randomly for each unit  $i$
4. Obtain the rank number  $r(t, x_{i,t}^*)$  for each individual  $x_{i,t}^*$  ( $m$  for the best, 1 for the worst) in the colony
5. Compute the covariance matrix  $\Sigma_t$  of  $m$  individuals of  $x_{i,t}^*$  of  $i=1,2,\dots,m$
6. Update the pheromone density  $\tau(t+1, x)$  according to Eq. 5
7. Sample new individual  $x_{i,t+1}^{new}$  according to Eq. 7 for  $i=1,2,\dots,m$
8. Apply perturbation to  $x_{i,t+1}^{new}$  with rate of  $P_{rate}$  for  $i=1,2,\dots,m$
9. Compare  $x_{i,t+1}^{new}$  and  $x_{i,t}^*$  and set the best one as  $x_{i,t+1}^*$  for  $i=1,2,\dots,m$
10.  $t \leftarrow t+1$
11. If the termination criteria are met, terminate the algorithm. Otherwise, go to 4

**Fig. 3.** The pseudo-code of eAPS

**3.2.3 The computational complexity** Here, we consider the computational complexity of the eAPS. Let  $n$  be the problem size. First, let us consider the computational complexity of updating the pheromone intensity described in 3.2.1. Comparing  $x_{i,t+1}^{new}$  and  $x_{i,t}^*$  for  $m$  units is simply performed with  $O(m)$ . To give the rank number for each individual  $x_{i,t}^*$ , we use a quick sort with a complexity of  $O(m \times \log(m))$ . Computing the covariance matrix  $\Sigma_t$  of  $m$  individuals is performed with  $O(m \times n^2)$ . Next, the computational complexity of the three samplings to generate  $m$  individual described in 3.2.2 is as follows: The iteration sampling is simply performed with  $O(t \times m)$  for  $t < H$  or  $O(H \times m)$  for  $t \geq H$ . The rank sampling is also simply performed with  $O(m^2)$ . Cholesky decomposition is performed with  $O(n^3)$  and the sampling after the Cholesky decomposition is  $O(m \times n^2)$ .

Thus the computation time of the algorithm is mainly occupied by computing the covariance matrix ( $O(m \times n^2)$ ), the Cholesky decomposition ( $O(n^3)$ ) and the sampling after the Cholesky decomposition ( $O(m \times n^2)$ ). However, since in real-world parameter optimization a function evaluation usually needs a much larger computational time compared to the evolutionary algorithm run time, the number of function evaluations is the more critical issue. In Section 4, we mainly evaluate eAPS using the number of function evaluations.

## 4 Experimental Study

In this section, the search capability and the characteristics of eAPS are studied in comparison with APS using test functions which are commonly used in the evolutionary computation community.

#### 4.1 Experimental Methodology

We used the following four test functions: the Ellipsoidal function ( $F_{Ellipsoidal}$ ) [19], the Ridge function ( $F_{Ridge}$ ), the Rosenbrock function ( $F_{Rosenbrock}$ ), and the Rastrigin function ( $F_{Rastrigin}$ ).

$$F_{Ellipsoidal} = \sum_{i=1}^n i x_i^2, \quad (-3.12 \leq x_i < 7.12) \quad (13)$$

$$F_{Ridge} = \sum_{i=1}^n \left( \sum_{j=1}^i x_j \right)^2, \quad (-44 \leq x_i < 84) \quad (14)$$

$$F_{Rosenbrock} = \sum_{i=2}^n (100(x_1 - x_i^2)^2 + (x_i - 1)^2), \quad (-2.048 \leq x_i < 2.048) \quad (15)$$

$$F_{Rastrigin} = 10n + \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i)), \quad (-3.12 \leq x_i < 7.12) \quad (16)$$

$F_{Ridge}$  has a linkage among variables.  $F_{Rosenbrock}$  has a strong linkage among variables.  $F_{Ellipsoidal}$  and  $F_{Rastrigin}$  have no linkage among variables.  $F_{Rosenbrock}$ ,  $F_{Ellipsoidal}$ , and  $F_{Ridge}$  are unimodal, and  $F_{Rastrigin}$  is multimodal.  $F_{Ellipsoidal}$ ,  $F_{Ridge}$ , and  $F_{Rastrigin}$  have their global optima at  $(0, 0, \dots, 0)$ . To avoid sampling bias as discussed in [19], search space definitions are altered from the original definitions.

We evaluated the algorithms by measuring  $\#OPT$  (the number of runs in which algorithms succeeded in finding the global optimum) and  $MNE$  (the mean number of function evaluations to find the global optimum in those runs where it did find the optimum). Problem size  $n = 20$  was used for all test functions. We assumed the solution to be successfully detected if the functional value was within  $n \times 10^{-6}$  of the actual optimum value. 20 runs were performed in each setting. Each run continued until the global optimum was found or a maximum of 500,000 evaluations was reached. There are four common parameters for both eAPS and APS:  $m$ ,  $\alpha$ ,  $\beta$ , and  $\rho$ . APS contains an extra parameter  $e$  which determines the number of best individuals passed to the next iteration by  $e \times m$  (see Fig. 1). In all experiments, the same unit size of  $m = 120$  was used. Perturbation was applied to each solution with  $P_{rate} = 0.0005$  by adding a randomly generated number from a zero-mean, unit-normal distribution. For the value of  $H$ ,  $H = 100$  was used. Here the remaining parameters were set as follows. For the APS, the following parameters were used in the experiments:  $\beta = 0.6$ ,  $\alpha = 4$ ,  $\rho = 0.9$ , and  $e = 0.1$ . For eAPS, the following parameters were used in the experiments:  $\beta = 1.0$ ,  $\alpha = 32$ , and  $\rho = 0.2$ , except for the value of  $\beta$  of  $F_{Rastrigin}$ . For  $F_{Rastrigin}$ ,  $\beta = 0.6$  was used. These values were obtained by tuning, using test function  $F_{Rosenbrock}$ .  $F_{Rosenbrock}$  was chosen because it is one of the hardest problems to solve due to a strong linkage among variables. For a comparison with a real-coded GA (RGA), the results were also compared to the results obtained with SPX crossover [20], a typical crossover operator for RGAs, with the population size tuned to 300. Parameters for SPX were also tuned using  $F_{Rosenbrock}$ .

## 4.2 Results

Table 1 shows the results of eAPS, APS, and RGA using the parameter values described in Section 4.1. Among these three, eAPS showed the best results with the smallest values of  $MNE$ . For examples, values of  $MNE$  on  $F_{Rosenbrock}$  are 61,768.7, 126,994.4, and 255,483.5 for eAPS, APS, and RGA, respectively. Values of  $\#OPT$  on  $F_{rastrigin}$  are 20, 16, and 17 for eAPS, APS, and RGA, respectively. ( $Time$  indicates the average time for successful runs in seconds on a 2.8G Hz Pentium 4 with 512MB main memory. The code is written in Java).

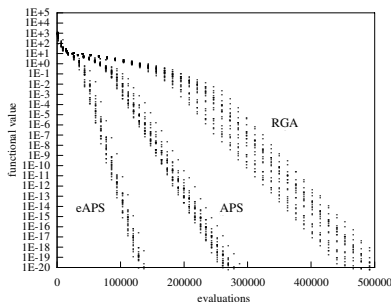
Fig. 4 shows the convergence processes of eAPS, APS, and RGA on  $F_{Rosenbrock}$ . Each dot sequence indicates a change of functional values over 20 runs with each algorithm. We can see that eAPS converged more rapidly than APS and RGA.

In addition to examining the performance results with fixed parameter values, we analyzed the sensitivity of eAPS and APS to changing parameters  $\rho$  and  $\alpha$ . To make the analysis feasible, the sensitivity of eAPS and APS with respect to only one parameter at a time is analyzed, while all the remaining parameters being kept constant.

Fig. 5 shows the variation of  $MNE$  and  $\#OPT$  with  $\rho$  for the function  $F_{Ellipsoidal}$  (unimodal, no linkage among variables) and  $F_{Rosenbrock}$  (unimodal, strong linkage among variables). The first figure shows the variation with eAPS and the second one with APS. To explore the effect of the evaporation coefficient  $\rho$ , we varied the value of  $\rho$  over the ranges of  $[0, 0.9]$  for eAPS and  $[0.5, 0.95]$  for APS. We can see that eAPS is robust to the variation of  $\rho$ , but APS is very sensitive to this value. In eAPS, the best individuals are maintained in each unit. This works as a kind of memory of past searches. This is the reason why eAPS is robust to variation of  $\rho$ . However, on  $F_{Rosenbrock}$  which has a strong linkage among variables,  $\rho$  still plays an important role. On the other hand, on  $F_{Ellipsoidal}$  which has no linkage among variables,  $\rho = 0$  showed the best performance. In APS, the appropriate choice of  $\rho$  value becomes important for APS to perform well.

**Table 1.** Results with default values

Algorithm	$F_{Ellipsoidal}$	$F_{Ridge}$	$F_{Rosenbrock}$	$F_{Rastrigin}$
eAPS	$\#OPT$	20/20	20/20	20/20
	$MNE$	<b>35927.7</b>	<b>43043.5</b>	<b>61768.7</b>
	$STD$	1137.9	1510.7	4169.6
	$Time(sec)$	1.8	1.4	2.7
APS	$\#OPT$	20/20	20/20	16/20
	$MNE$	70001.4	83730.6	126994.4
	$STD$	1521.5	1539.9	7325.7
	$Time(sec)$	3.1	3.8	5.5
RGA	$\#OPT$	20/20	20/20	17/20
	$MNE$	113076.9	137996.0	255483.5
	$STD$	1716.2	137996.0	21392.3
	$Time(sec)$	8.5	7.8	10.6



**Fig. 4.** Convergence processes on  $F_{Rosenbrock}$

To see the effect of the parameter  $\alpha$ , which adjusts the relative importance of rank in determining the pheromone emitted by an individual, we tested  $\alpha$  in the range of  $[4, 64]$  on the same functions (Fig. 6). With larger values of  $\alpha$ , higher ranked individuals emit pheromone at increasing rates. With larger values of  $\alpha$ , the performance of eAPS increased, as seen in the results with  $F_{Ellipsoidal}$ . However, as seen in the results with  $F_{Rosenbrock}$ , which has a strong linkage among variables, larger  $\alpha$  values affect convergence. Again in APS, varying  $\alpha$  has a strong effect on performance, especially on  $F_{Rosenbrock}$ .

In the above experiment, we analyzed the algorithms with problem size ( $n$ ) fixed at 20. Here we focus on the scale-up behavior of eAPS and APS using function  $F_{Rosenbrock}$  with the number of variables increasing from 10 to 40 with a step size of 10. We ran the experiment for each number of variables ( $n$ ) increasing the values of  $m$  starting at 100 with a step size of 20 until the optimal solution was obtained 5 times in 5 runs. The values of the other control parameters were the same as those described in Section 4.1. Fig. 7 shows changes of  $MNE$  and  $m$ . The  $MNE$  increased at rate of almost  $O(n^{2.6})$  and  $m$  increased almost linearly. However, we need to undertake a more intensive study on the scaling behavior of the algorithms before drawing firm conclusions.

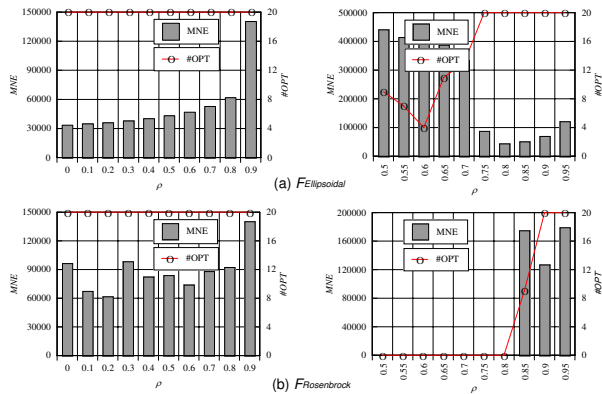


Fig. 5. Variation of  $MNE$  and  $\#OPT$  with  $\rho$

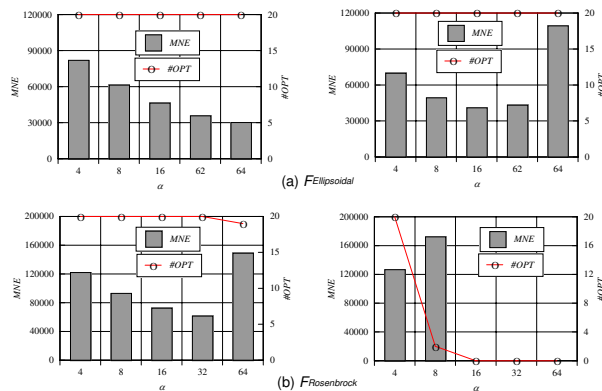


Fig. 6. Variation of  $MNE$  and  $\#OPT$  with  $\alpha$

## 5 Conclusions

In this paper, we have proposed an enhanced Aggregation Pheromone System (eAPS), which uses a colony model divided into units. Experimental results showed that eAPS has higher performance than the previously proposed APS. It has also been shown that in eAPS the parameter settings are more robust.

Although the results obtained with eAPS on standard test problems are promising, we do not claim that eAPS is the most efficient algorithm for solving the test problems. However, it provides an alternative approach to the use of probabilistic model building and sampling in evolutionary algorithms [21] for solving real-valued problems. There are many important topics for future research:

1. It is important to identify classes of problems for which eAPS outperforms advanced evolutionary algorithms in continuous domains, such as, CMA-ES [22] and PCX [19].
2. Since the model used in eAPS to sample new individuals can contain multiple attractors, another interesting area for future research is to extend eAPS to deal with multi-objective problems in the continuous domain.

## Acknowledgements

The author would like to acknowledge Dr. Marco Dorigo, the research director of the IRIDIA Lab. at the Université Libre de Bruxelles, for his advice on improving APS. This research is partially supported by the Ministry of Education, Culture, Sports, Science and Technology of Japan under Grant-in-Aid for Scientific Research number 16500143.

## References

1. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: Optimization by a colony of cooperating agents. *IEEE Trans. on SMC-Part B* **26**(1) (1996) 29–41
2. Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. on EC* **1**(1) (1997) 53–66
3. Dorigo, M., Stützle, T.: *Ant Colony Optimization*. MIT press, Massachusetts (2004)
4. Stützle, T., Hoos, H.: Max-min ant system. *Future Generation Computer Systems* **16**(9) (2000) 889–914

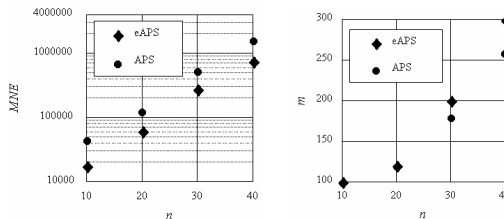


Fig. 7. Scale-up behavior

5. Maniezzo, V.: Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. Research Report CSR 98-1 (1998)
6. Forsyth, P., Wren, A.: An ant system for bus driver scheduling. Proc. of the 7th Int. Workshop on Computer-Aided Scheduling of Public Transport (1997)
7. Bullnheimer, B., Hartl, R.F., Strauss, C.: Applying the Ant System to the Vehicle Routing Problem. Meta-heuristics: Advances and trends in local search paradigms for optimization, voss, s., et al (eds.) edn. Kluwer (1999)
8. Bilchev, G., Parmee, I.C.: The ant colony metaphor for searching continuous design spaces. Proc. of the AISB Workshop on Evo. Comp. (1995) 24–39
9. Mathur, M., Karle, S.B., Priye, S., Jyaraman, V.K., Kulkarni, B.D.: Ant colony approach to continuous function optimization. Ind. Eng. Chem. Res **39** (2000) 3814–3822
10. Wodrich, M., Bilchev, G.: Cooperative distribution search: the ant't way. Control Cybernetics **3** (1997) 413–446
11. Monmarch'e, N., Venturini, G., Slimane, M.: On how pachycondyla apicalis ants suggest a new search algorithm. Future Generation Computer Systems **16**(8) (2000) 937–946
12. Dr'eo, J., Siarry, P.: A new ant colony algorithm using the heterarchical concept aimed at optimization of multim minima continuous functions. Proc. of the Third Int. Workshop on Ant Algorithms (ANTS 2002) (2002) 216–221
13. Pourtaqdoust, S.H., Nobahari, H.: An extension of ant colony system to continuous optimization problems. Proc. of Fourth Int. Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS 2004 ) (2004) 294–301
14. Socha, K.: Aco for continuous and mixed-variable optimization. Proc. of Fourth Int. Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS 2004) (2004) 25–36
15. Tsutsui, S.: Ant colony optimisation for continuous domains with aggregation pheromones metaphor. Proc. of the fifth Int. Conf. on Recent Advances in Soft Computing (2004) 207–212
16. Tsutsui, S., Pelikan, M., Ghosh, A.: Performance of aggregation pheromone system on unimodal and multimodal problems. Proc. of the 2005 IEEE CEC (2005) 880–887
17. Bell, W.J., Car, R.T.: Chemical ecology of insects. Journal of Applied Ecology **22**(1) (1985) 299–300
18. Schatzman, M., Taylor, J., Schatzman, M.: Numerical Analysis: A Mathematical Introduction. Oxford Univ Press (2002)
19. Deb, K., Anand, A., Joshi, D.: A computationally efficient evolutionary algorithm for real-parameter optimization. Evolutionary Computation **10**(4) (2002) 371–395
20. Higuchi, T., Tsutsui, S., Yamamura, M.: Theoretical analysis of simplex crossover for real-coded genetic algorithms. Proc. of the PPSN VI (2000) 365–374
21. Pelikan, M., Goldberg, D.E., Lobo, F.: A survey of optimization by building and using probabilistic models. Computational Optimization and Applications **21**(1) (2002) 5–20
22. Hansen, N., Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. Proc. of the 1996 CEC (1996) 312–317