ACO for Continuous and Mixed-Variable Optimization

Krzysztof Socha

IRIDIA – Université Libre de Bruxelles – Brussels, Belgium
ksocha@ulb.ac.be

Abstract. This paper presents how the Ant Colony Optimization (ACO) metaheuristic can be extended to continuous search domains and applied to both continuous and mixed discrete-continuous optimization problems. The paper describes the general underlying idea, enumerates some possible design choices, presents a first implementation, and provides some preliminary results obtained on well-known benchmark problems. The proposed method is compared to other ant, as well as non-ant methods for continuous optimization.

1 Introduction

Optimization algorithms inspired by the ants' foraging behavior proposed by Dorigo in his PhD thesis in 1992 have been initially used for solving combinatorial optimization problems. They have been eventually formalized into the framework of the Ant Colony Optimization (ACO) metaheuristic [7]. ACO has proven to be an efficient and versatile tool for solving various combinatorial optimization problems. Several versions of ACO have been proposed, but they all follow the same basic ideas:

- search performed by a population of *individuals*, i.e. simple independent agents,
- incremental construction of solutions,
- probabilistic choice of solution components based on stigmergic information,
- no direct communication between the individuals.

Since the emergence of ant algorithms as an optimization tool, some attempts were also made to use them for tackling continuous optimization problems. However, at the first sight, applying the ACO metaheuristic to continuous domain was not straightforward. Hence, the methods proposed often drew inspiration from ACO, but did not follow exactly the same methodology.

Up to now, only a few ant approaches for continuous optimization have been proposed in the literature. The first method – called Continuous ACO (CACO) – was proposed by Bilchev and Parmee [2] in 1995, and also later used by some others [17, 12]. Other methods include the API algorithm by Monmarché [13], and Continuous Interacting Ant Colony (CIAC), proposed by Dréo and Siarry [9, 8].

M. Dorigo et al. (Eds.): ANTS 2004, LNCS 3172, pp. 25–36, 2004.

Although both CACO and CIAC claim to draw inspiration from the ACO metaheuristic, they do not follow it closely. All the algorithms add some additional mechanisms (e.g. direct communication – CIAC and API – or nest – CACO) that do not exist in regular ACO. They also disregard some other mechanisms that are otherwise characteristic of ACO (e.g. stygmergy – API – or incremental construction of solutions – all of them). CACO and CIAC are dedicated strictly to continuous optimization, while API may also be used for discrete problems.

Contrary to those earlier approaches, this paper presents a way to extend a generic ACO to continuous domains without the need to make any major conceptual changes. Such extended ACO, due to its closeness to the original formulation of ACO, provides an additional advantage – the possibility of tackling mixed discrete-continuous optimization problems. In other words, with ACO it should be now possible to consider problems where some variables are discrete and others are continuous.

The reminder of the paper is organized as follows. Section 2 presents the idea and enumerates the possible design choices. Section 3 provides a short discussion of the proposed solution with regard to other methods for continuous and mixed-variable optimization. Section 4 presents the choices made for the first implementation and compares some initial results with those obtained by competing methods. Finally, Sec. 5 presents the conclusions and future work plans.

2 ACO Extended to Continuous Domain

When ACO is used for combinatorial optimization problems, ants construct solutions incrementally. Each ant starts with an empty solution S^0 and at each construction step i a component of the solution is added. The definition of a *solution component* depends on the problem tackled. In case of the popular example of Traveling Salesman Problem (TSP), a component of the solution is a city that is added to a tour. For other problems the solution components may be defined differently.

In order to choose, which of the available solution components $\boldsymbol{C}^i$ should be added to the current partial solution S^i, a probabilistic choice is made. This decision is usually influenced by amount of *pheromone* τ associated with available choices, and by heuristic information about the problem. Without the loss of generality, we focus on a case when no heuristic information is used. The probability of choosing a solution component $c \in \boldsymbol{C}^i$ at step i in iteration t, assuming that the partial solution constructed so far is S^i, is a normalized pheromone value associated with this component:

$$p_{S^i c}(t) = \frac{\tau_{S^i c}(t)}{\sum_{j \in \boldsymbol{C}^i} \tau_{S^i j}(t)} \tag{1}$$

Hence, in case of combinatorial optimization problems, at each construction step the ants make a probabilistic decision according to some discrete probability distribution.

Algorithm 1 Ant Colony Optimization extended to continuous domain

input: An objective function $\mathbb{R} \ni f(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^n$
$\tau^i \leftarrow$ initial probability distribution $P^i(x^i),\ i \in \{1..n\}$
while (stop condition not met) **do**
  {iterate through all m ants}
  for $a = 1$ **to** m **do**
    {construction process of ant a}
    $s^0 \leftarrow \emptyset$
    for $i = 1$ **to** n **do**
      choose value x^i randomly according to probability distribution $P^i(x^i)$
      $s^i \leftarrow s^{i-1} \cup \{x^i\}$
    end for
  end for
  $s_{\mathrm{I\,best}} \leftarrow$ iteration best solution
  $s_{\mathrm{G\,best}} \leftarrow$ best of $s_{\mathrm{I\,best}}$ and previous global best $s_{\mathrm{G\,best}}$
  $\tau \leftarrow$ pheromone updated based on one or more solutions found
end while
output: Best solution found $s_{\mathrm{G\,best}}$.

In case of continuous optimization problems, the domain changes from discrete to continuous. The logical adaptation would be to also move from using the discrete probability distribution to a continuous one – the Probability Density Function (PDF). Instead of choosing at step i a component $c \in \boldsymbol{C}^i$, the ants would generate a random number according to a certain PDF $P^i(x^i)$.

More formally, the proposed ACO algorithm extended to continuous domain is presented in Alg. 1. Clearly, the general structure does not differ from a generic ACO, but it is extended to handle continuous variables. The probability distributions $P^i(x^i)$ may be either discrete or continuous.

2.1 Probability Density Function (PDF)

One of the most popular functions that is used as PDF for estimating distributions is the normal (or gaussian) function:

$$g(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2}$$

The normal PDF has some clear advantages, such as a fairly easy way of generating random numbers according to it, but it also has some disadvantages. A single normal PDF is not able to describe a situation where two disjoint areas of the search space are promising, as it only has one maximum.

Due to this fact, we used a distribution based on the normal PDF, but slightly enhanced – a mixture of normal kernels. Similar constructs have been used before [3], but not exactly in the same way. We define it as a weighted sum of several normal PDFs, and denote it as G:

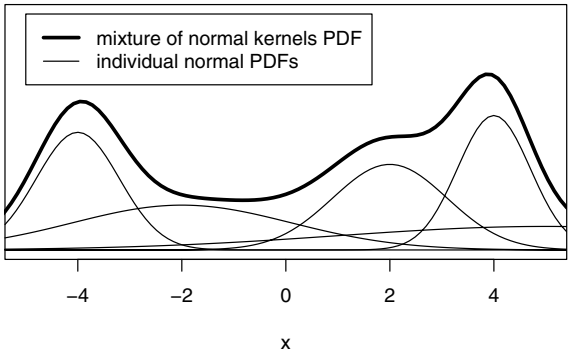

Fig. 1. Example of five normal PDFs and their sum – the resulting mixture of normal kernels – on the interval $(-5, 5)$

$$P(x) = G(x, \boldsymbol{\omega}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = \sum_{j=1}^{k} \omega_j \cdot g(x, \mu_j, \sigma_j) \tag{3}$$

where $\boldsymbol{\omega}$ is the vector of weights associated with the components of the mixture, $\boldsymbol{\mu}$ is the vector of means, and $\boldsymbol{\sigma}$ is the vector of standard deviations. The dimensions of all those vectors are equal to the number of normal PDFs constituting the mixture. For convenience we will use parameter k to describe this number $\dim \boldsymbol{\omega} = \dim \boldsymbol{\mu} = \dim \boldsymbol{\sigma} = k$.

Such a distribution allows for reasonably easy generation of random numbers accoding to it, and yet it provides a much increased flexibility in the possible shape. An example of how such a mixture may look like is presented in Fig. 1.

For the remaining part of this paper we will use the notation $P^i(x^i)$ to indicate the i-th mixture of normal kernel PDFs: $P^i(x^i) = G(x^i, \boldsymbol{\omega}^i, \boldsymbol{\mu}^i, \boldsymbol{\sigma}^i)$, and $P^i_j(x^i)$ to indicate the j-th single normal PDF: $P^i_j(x^i) = g(x^i, \mu^i_j, \sigma^i_j)$ being part of the i-th mixture. Therefore:

$$P^i(x^i) = \sum_{j=1}^{k^i} \omega^i_j \cdot P^i_j(x^i) \tag{4}$$

Additionally, we will use simply the term *mixture* to refer to a mixture of normal kernel PDFs.

2.2 Solution Construction

The construction of solutions in ACO for continuous domain is done in principle in the same manner as in the case of regular ACO. At each step i, each of the ants chooses a component of the solution based on the probability distribution. If the component is to be chosen based on discrete probability distribution, it is done exactly the same way as in the case of regular ACO.

In case of continuous domain, a component is a value x^i of a single dimension of the solution $\boldsymbol{x} \in \mathbb{R}^n$. Rather than using discrete distribution, the continuous PDF $P^i(x^i)$ is used as defined in (3). At step i, an ant generates a random number according to the i-th mixture $P^i(x^i)$. This is accomplished in two stages. First, an ant chooses probabilistically a single normal PDF $P^i_j(x)$ from the mixture $P^i(x^i)$ as in (4), with probability p^i_j proportional to ω^i_j:

$$p^i_j = \frac{\omega^i_j}{\sum_{l=1}^{k^i} \omega^i_l} \tag{5}$$

Following that, an ant generates a random number according to the chosen $P^i_j(x^i)$. This may be done using a random number generator that is able to generate random numbers according to a parameterized normal distribution, or using a uniform random generator and (for instance) the Box-Muller method [4].

2.3 Pheromone Maintenance

Each mixture $P^i(x^i)$ representing the i-th pheromone distribution is described by a triplet of vectors $(\boldsymbol{\omega}^i, \boldsymbol{\mu}^i, \boldsymbol{\sigma}^i)$ of equal dimension k^i. The larger the k^i, the more complex distributions may be described by this PDF. In particular, such a PDF may have at the most k^i maxima.

Initialization. Initially the pheromone distribution $P^i(x^i)$ must consist of at least one (it could be more) normal PDF $P^i_j(x^i)$. A reasonable choice must be made. If no prior information is given about the problem, it is reasonable to start with a rather uniform distribution over the search domain (a, b). This is not quite achievable using the mixture of normal kernel PDFs, but reasonable examples include a single normal PDF[1] $P^i_1(x^i)$:

$$P^i(x^i) = P^i_1(x^i) = g\left(x^i, \frac{a+b}{2}, \frac{b-a}{2}\right) \tag{6}$$

or a set of k^i normal distributions $P^i_j(x^i)$ with uniformly distributed means:

$$P^i(x^i) = \sum_{j=1}^{k^i} \frac{1}{k^i} \cdot g\left(x^i, a + (2j-1)\frac{b-a}{2k^i}, \frac{b-a}{2k^i}\right) \tag{7}$$

In some cases, in particular when the algorithm uses frequent restarts, the distribution $P^i(x^i)$ may be initialized with a randomized version of the one presented in (7). In such a case, one or more parameters of the distribution may be randomized: (i) standard deviations, (ii) means, (iii) weights, or (iv) number k^i of the normal PDFs.

[1] Note that it is necessary in such a case that $\omega^i_1 = 1$.

Update. Pheromone update is a process of modifying the probability distribution used by the ants during the construction process, so that it can guide the ants towards better solutions. This process traditionally consists of two actions: (i) reinforcing the probability of the choices that lead to good solutions – *positive update*, and (ii) decreasing probability of other choices (i.e. *forgetting* bad solutions) – *negative update*.

In ACO for continuous domain, the positive update may be accomplished by incorporating in the mixture $P^i(x^i)$, an additional normal PDF $P^i_j(x^i)$ for each solution used for the update. The mean μ^i_j of this new distribution should be equal to the value of the solution component x^i used for updating the probability distribution $P^i(x^i)$. The values of ω^i_j and σ^i_j may be chosen based on the state of the search, or the quality of the solution used for the update. The number of normal PDFs k^i constituting the mixture $P^i(x^i)$ should be appropriately increased: $k^i \leftarrow k^i + o$, where o is the number of solutions used for the update. For each dimension $i \in \{1..n\}$ the update is given by:

$$\begin{cases} P^i(x^i) \leftarrow P^i(x^i) + \sum_{j=1}^{o} P^i_j(x^i) \\ k^i \leftarrow k^i + o \end{cases} \tag{8}$$

Negative update is usually done in traditional ACO through pheromone evaporation [7]. However, also other alternative solutions have been presented in the literature. For instance, in Population-Based ACO [10]–the pheromone is added or removed from the pheromone matrix, as the individual solutions enter or are being removed from the population.

ACO for continuous domain allows for significant flexibility in the way the negative update is accomplished. In the following paragraphs we describe three example methods of achieving it. Others may also be possible.

One of the most obvious methods for negative update is the opposite of the positive update method presented in (8). Just as any new normal PDF $P^i_j(x^i)$ may be added to the mixture $P^i(x^i)$, in the same manner any existing one may be removed. More formally the process of removing a set of $\boldsymbol{O} \subset \{1..k^i\}$ normal PDFs for each dimension $i \in \{1..n\}$ may be presented as follows:

$$\begin{cases} P^i(x^i) \leftarrow P^i(x^i) - \sum_{j \in \boldsymbol{O}} P^i_j(x^i) \\ k^i \leftarrow k^i - |\boldsymbol{O}| \end{cases} \tag{9}$$

The second method for negative update is inspired directly by the pheromone evaporation in case of generic ACO. In case of a mixture of normal kernels PDF $P^i(x^i)$, it is possible to evaporate the vector of weights $\boldsymbol{\omega}^i$, for each dimension $i \in \{1..n\}$ and each normal PDF $j \in \{1..k^i\}$ according to evaporation rate ρ:

$$\omega^i_j \leftarrow (1 - \rho) \cdot \omega^i_j \tag{10}$$

The third method that may be considered, exploits the properties of the probability distribution used – the mixture of normal kernel PDFs. Assuming

that the positive update is done as presented in (8), each new normal PDF added to the mixture is based on some point that identifies a promising area of the search space. The idea of negative update is that as time passes, the old promising areas may become less promising. One of the ways of expressing this fact is, in the case of the normal PDF, the increase of the standard deviation σ. As this does not mean that the particular normal PDF will be less probable, but rather that the numbers generated according to this distribution will have more spread, we call this type of negative update *dissolving* instead of evaporation. The idea is then to increase the standard deviation σ_j^i of normal PDF $P_j^i(x^i)$ for each dimension $i \in \{1..n\}$ and for each normal PDF $j \in \{1..k^i\}$, with each iteration by:

$$\sigma_j^i \leftarrow \gamma \cdot \sigma_j^i \tag{11}$$

where γ is the parameter describing the rate of dissolving.

Obviously, the three methods proposed here for negative update of the probability distribution may be combined. In particular, the method presented in (9) may be combined with the other two in order to maintain a reasonable number of normal PDFs within the mixture. Otherwise, the size of the mixture may become too large to be handled easily. Ideas based on $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System [16] may be used for removing for instance only those normal PDFs whose weight ω dropped below a certain minimal value, or whose standard deviation σ exceeded a certain maximal threshold.

3 Discussion

The ACO algorithm presented in this paper may be studied from at least two different points of view. It may be used either as a method of solving continuous (or *global*) optimization problems, or as a method of solving mixed-variable (discrete and continuous) optimization problems.

3.1 Continuous Domain

ACO as a continuous optimization algorithm is part of a rather large family of algorithms for continuous optimization. For these types of problems, a number of methods have been proposed in the literature. They include some ant-related methods [2, 13, 9] already briefly presented in Sec. 1, but also many others. Many optimization algorithms have been originally developed for combinatorial optimization and only afterwards adapted also to the continuous case. Examples include the Continuous Genetic Algorithm (CGA) [6], Enhanced Simulated Annealing (ESA) [15], or Enhanced Continuous Tabu Search (ECTS) [5].

There are also other methods that – similarly to ACO – explicitly use some notion of probability estimation. Examples of the latter include the Iterated Density Estimation Algorithm (IDEA), or PBIL–Population-Based Incremental Learning. Similarly to ACO, they have been initially used for combinatorial optimization, and only later also adapted to handle continuous domains [3, 18].

ACO for continuous domain is similar to continuous PBIL or IDEA in the same sense, as generic ACO is similar to their discrete counterparts. It explicitly uses an estimation of probability distribution in order to find promising areas for the search. ACO differs from almost all of the methods mentioned here in the sense that it does not simply choose and evaluate the solutions, but builds them incrementally. Only PBIL uses similar incremental approach.

3.2 Mixed Variable Domain

ACO as an algorithm for optimization of mixed-variable problems does not have too many competitors at the moment. None of the other ant-related methods allows the intuitive handling of mixed-variable problems. There are few examples in the literature of other types of algorithms that explicitly do that. They include the Mixed Bayesian Optimization Algorithm (MBOA) [14], Pattern Search Algorithms [1], and the Bell-Curve Genetic Algorithm [11].

Since the mixed-variable optimization is not yet a very popular subject, there are not many benchmark problems available that would allow to properly test such an algorithm. Mixed-variable optimization remains nonetheless an interesting area of research. The most obvious field where it may be used is optimization of parameters. This could mean parameters of any physical process, or an algorithm's parameters – any case where some of the parameters may only assume discrete and others continuous values from a certain range. A practical example of such a mixed variable problem is the design of a thermal insulation system, as presented in [1].

4 Initial Results

4.1 Implementation

The extended version of ACO algorithm presented here may be used for both continuous and mixed-variable optimization problems. As a proof of concept we have implemented ACO for continuous optimization problems. Many benchmarks are available for these types of problems, and it is easy to compare the results of ACO with those obtained by other methods.

We have implemented ACO allowing optimization of multidimensional continuous functions on a given range and to a given accuracy. As input, the algorithm takes the following: the function to be optimized $f(\boldsymbol{x}) \in \mathbb{R}$, the search domain $\boldsymbol{x} \in (a, b)^n \subset \mathbb{R}^n$ – where interval (a, b) is assumed to be the same for all n dimensions, the known optimum[2] s_o, the accuracy ϵ, and a set of parameters, such as the number of ants m and the size of the mixtures k. As output, the algorithm presents the best solution found and the number of function evaluations performed.

[2] As this is proof of concept algorithm, the known optimum was used to test the performance of the algorithm against other methods.

The algorithm uses a different mixture of normal kernel PDFs for each of the n dimensions. Each mixture consists of the same number $k = k^{1..n}$ of normal PDFs. The solution construction is done exactly as presented in Sec. 2.2. The pheromone update is done in the following way: At each iteration, the *iteration best* solution is used for pheromone update. Positive update is done according to (8), and negative update according to (9) – only one (the oldest) normal PDF is removed. Since both actions are performed at the same time, the number k of normal PDFs in the mixture does not change. An additional improvement in the algorithm's performance was achieved by adding a simple elitist strategy. The normal PDF associated with the *global best* solution is never removed. If it is also the oldest one, then the second oldest normal PDF is rather removed.

The pheromone distribution for each variable x^i is initialized with a slightly randomized version of the one presented in (7). The vector of means $\boldsymbol{\mu}$ is randomly selected from the interval (a, b) using a uniform random number generator.

At each iteration of the algorithm, m ants construct the m solutions to the problem. A different mixture $P^i(x^i)$ is used when choosing each component x^i of the solution. Due to the nature of the PDF used, it is possible that some of the generated solutions $\boldsymbol{S}^c$ will fall outside of the search domain $\boldsymbol{x} \in (a, b)^n \subset \mathbb{R}^n$. Those solutions are dropped before being evaluated. All other solutions are evaluated, and the *iteration best* solution $s_{\mathrm{I\,best}}$ is chosen. The oldest component $P^i_j(x^i)$ of each mixture is removed, and a new component is added. The new component PDF takes its mean from the value of the respective solution variable: $\mu^i = x^i_{s_{\mathrm{I\,best}}}$.

The standard deviation σ^i of the normal PDFs $P^i_j(x^i)$ used for the update, is chosen adaptively based on the index of current iteration c and the solutions found in this iteration $s_{1..m} = (x^1, .., x^n)_{1..m} \in \boldsymbol{S}^c$:

$$\sigma^i = \max\left(\frac{\max\left(x^i_{1..m}\right) - \min\left(x^i_{1..m}\right)}{\sqrt{c}}, \epsilon\right) \tag{12}$$

At each iteration, if the $s_{\mathrm{I\,best}}$ is better than the $s_{\mathrm{I\,best}}$, the latter is replaced. The stopping criterion is the required accuracy. The new global best solution $s_{\mathrm{G\,best}}$ is compared to the known optimum s_o. The algorithm stops if the following condition is met [6, 8]:

$$|s_{\mathrm{G\,best}} - s_o| < \epsilon_{\mathrm{rel}} \cdot s_o + \epsilon_{\mathrm{abs}} \Rightarrow \mathrm{STOP} \tag{13}$$

where $\epsilon_{\mathrm{rel}} = \epsilon_{\mathrm{abs}} = \epsilon$.

4.2 Results

The preliminary results presented here are based on 100 independent runs of the ACO algorithm on each of the test functions. The parameters used for tackling each test function are presented in Tab. 1. In order to have comparable results, the accuracy ϵ was chosen based on the results of the other algorithms published in the literature. The other parameters were chosen empirically based on some limited experimentation – about 10 configurations of parameters were tried for

Table 1. ACO parameters used for solving different benchmark problems: m–number of ants, k–number of normal PDFs per each of n dimensions, and ϵ–required accuracy

Test Function	d	m	k	ϵ
Sphere Model (SM)	6	8	3	$1 \cdot 10^{-4}$
Goldstein and Price (GP)	2	6	4	$1 \cdot 10^{-4}$
Rosenbrock (R_2)	2	30	8	$3 \cdot 10^{-3}$
Zakharov (Z_2)	2	8	4	$1 \cdot 10^{-4}$
Hartmann ($H_{3,4}$)	3	12	5	$1 \cdot 10^{-3}$

Table 2. Comparison of average number of function evaluations until the algorithm stop, on different benchmark problems. Results for some of the algorithms were not available – hence some entries are missing. The brackets indicate that the results are based on the runs with a fixed number of evaluations

		Other Ant Methods			Non-Ant Methods		
$f(\boldsymbol{x})$	ACO	CACO [17]	API [13]	CIAC [8]	CGA [6]	ECTS [5]	ESA [15]
SM	695	22050	[10000]	50000	750	338	-
GP	364	5330	-	23391	410	231	783
R_2	2905	6842	[10000]	11797	960	480	796
Z_2	401	-	-	-	620	195	15820
$H_{3,4}$	457	-	-	-	582	548	698

each test function. The results obtained by the other methods found in the literature are presented in Tab. 2.

The comparison presented here may be used only as an indication of the potential of ACO, and not as an exhaustive comparison of continuous optimization methods. The results found in the literature have been obtained under different conditions (e.g. slightly different stopping criteria). Also, the simple average number of evaluations does not fully describe an algorithm's performance. Other measures of the performance may be important, such as variance, stability, accuracy, or even the ease of the implementation.

It is clear that when considering only the number of function evaluations, ACO (for the limited number of cases presented here) is better than any of the other ant-related methods. However, more detailed analysis on a wider sample of test functions will have to be performed in order to add statistical significance to this claim. Considering the other continuous optimization methods, ACO's performance is similar. Again, larger sets of benchmarks and unified evaluation methods would have to be employed to indicate particular advantages or disadvantages of ACO.

Due to the space limitations, the definitions of the test functions used for performance evaluation are not provided in the paper. They may be found in the literature [8, 6, 5], and also online[3], along with the source code (in R) of the ACO used to obtain the results presented in this paper.

[3] `http://iridia.ulb.ac.be/~ksocha/extaco04.html`

5 Conclusions

We have shown how Ant Colony Optimization may be extended to continuous and mixed-variable optimization domains. This can be done without any major conceptual change to the original definition of ACO.

We have shown how such a new ACO algorithm may be designed, and what are the possible design choices. We have reviewed other methods used for both continuous and mixed-variable optimization from the literature. We have positioned ACO among them and explained how it differs from all other ant-related approaches to continuous optimization.

Finally, we have presented the first implementation of ACO for continuous problems, and shown that the method is at least competitive with the others found in the literature.

Our future work plans are twofold. On one side we would like to fine-tune the algorithm, so that its performance on continuous benchmark functions may be further improved. Also, as the initial tests of the algorithm were performed on problems that only had few dimensions, we plan to test the algorithm on higher-dimensional problems.

The second direction of possible research is a practical application of ACO to mixed-variable optimization. We plan to either find or create benchmarks that would allow the evaluation the performance of the algorithm on those types of problems. Also, if possible we will look for practical examples, where mixed-variable optimization may be used.

Acknowledgments

This research was supported by the "Metaheuristics Network", a Marie Curie Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-CT-1999-00106, and by the "ANTS" project, an "Action de Recherche Concertée" funded by the Scientific Research Directorate of the French Community of Belgium.

References

1. C. Audet and J. J. E. Dennis. Pattern Search Algorithms for Mixed Variable Programming. *SIAM Journal on Optimization*, 11(3):573–594, 2001.
2. G. Bilchev and I. C. Parmee. The Ant Colony Metaphor for Searching Continuous Design Spaces. In T. C. Fogarty, editor, *Proceedings of the AISB Workshop on Evolutionary Computation*, volume 993 of *LNCS*, pages 25–39. Springer-Verlag, Berlin, Germany, 1995.
3. P. A. N. Bosman and D. Thierens. Continuous Iterated Density Estimation Evolutionary Algorithms within the IDEA Framework. In M. Pelikan, H. Mühlenbein, and A. O. Rodriguez, editors, *Proceedings of OBUPM Workshop at GECCO-2000*, pages 197–200. Morgan-Kaufmann Publishers, San Francisco, CA, USA, 2000.
4. G. E. P. Box and M. E. Muller. A note on the generation of random normal deviates. *Annals of Mathematical Statistics*, 29(2):610–611, 1958.

5. R. Chelouah and P. Siarry. Enhanced Continuous Tabu Search. In S. Voss, S. Martello, I. H. Osman, and C. Roucairol, editors, *Meta-Heuristics Advances and Trends in Local Search Paradigms for Optimization*, chapter 4, pages 49–61. Kluwer Academic Publishers, Boston, MA, USA, 1999.
6. R. Chelouah and P. Siarry. A Continuous Genetic Algorithm Designed for the Global Optimization of Multimodal Functions. *Journal of Heuristics*, 6:191–213, 2000.
7. M. Dorigo and G. Di Caro. The Ant Colony Optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*. McGraw-Hill, New York, NY, USA, 1999.
8. J. Dréo and P. Siarry. Continuous Interacting Ant Colony Algorithm Based on Dense Heterarchy. *Future Generation Computer Systems*, to appear.
9. J. Dréo and P. Siarry. A New Ant Colony Algorithm Using the Heterarchical Concept Aimed at Optimization of Multiminima Continuous Functions. In M. Dorigo, G. D. Caro, and M. Sampels, editors, *Proceedings of the Third International Workshop on Ant Algorithms (ANTS'2002)*, volume 2463 of *LNCS*, pages 216–221. Springer-Verlag, Berlin, Germany, 2002.
10. M. Guntsch and M. Middendorf. A population based approach for ACO. In S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G. Raidl, editors, *Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2002: EvoCOP, EvoIASP, EvoSTim*, volume 2279, pages 71–80. Springer-Verlag, Berlin, Germany, 3-4 2002.
11. R. K. Kincaid, S. Griffith, R. Sykes, and J. Sobieszczanski-Sobieski. A Bell-Curve Genetic Algorithm for Mixed Continuous and Discrete Optimization Problems. In *Proceedings of 43rd AIAA Structures, Structural Dynamics, and Materials Conference*. AIAA, Denver, CO, USA, 2002.
12. M. Mathur, S. B. Karale, S. Priye, V. K. Jyaraman, and B. D. Kulkarni. Ant Colony Approach to Continuous Function Optimization. *Ind. Eng. Chem. Res.*, 39:3814–3822, 2000.
13. N. Monmarché, G. Venturini, and M. Slimane. On how *Pachycondyla apicalis* ants suggest a new search algorithm. *Future Generation Computer Systems*, 16:937–946, 2000.
14. J. Očenášek and J. Schwarz. Estimation Distribution Algorithm for Mixed Continuous-Discreet Optimization Problems. In *Proceedings of the 2nd Euro-Inernational Symposium on Computational Inteligence*, pages 227–232. IOS Press, Amsterdam, Netherlands, 2002.
15. P. Siarry, G. Berthiau, F. Durbin, and J. Haussy. Enhanced Simulated Annealing for Globally Minimizing Functions of Many Continuous Variables. *ACM Transactions on Mathematical Software*, 23(2):209–228, 1997.
16. T. Stützle and H. H. Hoos. $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.
17. M. Wodrich and G. Bilchev. Cooperative distributed search: the ant's way. *Control & Cybernetics*, (3):413–446, 1997.
18. B. Yuan and M. Gallagher. Playing in Continuous Spaces: Some Analysis and Extension of Population-Based Incremental Learning. In Sarker, R. *et al.*, editor, *Proceedings of Congress of Evolutionary Computation (CEC)*, pages 443–450, 2003.