Fitting of an Ant Colony approach to Dynamic Optimization through a new set of test functions

Walid Tfaili, Johann Dréo and Patrick Siarry

Université Paris XII Val-de-Marne, LISSI, E.A. 3956, 61 avenue du Général de Gaulle, 94010 Créteil, France
{tfaili, dreo, siarry}@univ-paris12.fr

February 14, 2007

Abstract

Real world problems are often of dynamic nature. They form a class of difficult problems that metaheuristics aim to solve. The goal is not only to attempt to find near-to-optimal solutions for a defined objective function, but also to track them in the search space. We will discuss in this article the dynamic optimization in the continuous case. Then we will present the experimentation on a battery of test functions, specially tuned for that purpose, of our ant colony algorithm, DHCIAC (Dynamic Hybrid Continuous Interacting Ant Colony).

Keywords: Dynamic optimization, ant colony, Nelder-Mead, particle swarm optimization, simplex, genetic algorithm.

1 Introduction

Metaheuristics are a class of stochastic methods that proved to be interesting for their versatility \[17\] \[11\]. They are generally used to solve combinatorial discrete NP-hard problems, but real world problems are often defined with real, or mixed integer/real parameters.

In the last few years, the literature shows an increasing interest for dynamic problems. Most commonly used metaheuristics are evolutionary algorithms, that can be applied to combinatorial problems. One promising approach in this field is embodied by the Ant Colony Optimization algorithms. Our concern in this paper is continuous dynamic hard optimization problems, solved by using an ant colony algorithm.

In section 2, we will shortly expose the related works. Our DHCIAC algorithm is described in detail in section 3. In section 4.1 the performance measurements found in the literature and used in this paper are listed. Results and comparison with other approaches are presented in section 5 using a new set of dynamic test functions. Then we conclude in section 6.

2 Metaheuristics for dynamic optimization

Most metaheuristics are sophisticated Monte-Carlo methods. The basic Monte-Carlo method is a random search algorithm which seeks the function optimum by generating a sample of solutions according to an uniform law. At the beginning of 90’s, a new research domain in distributed artificial intelligence has emerged, called swarm intelligence. It concerns the study of the utility of mimicking social insects for conceiving new algorithms. Computer science has used the concept of auto-organization and emergence, found in social insects societies, to define what is called swarm intelligence. The ant colony algorithm (based on the deposit and the evaporation of pheromone) and the particle swarm optimization algorithm (PSO) are only two methods among others inspired by nature. One of the
major advantages of swarm intelligence algorithms, such as the ant colony algorithms, is their flexibility in dynamic environments. Nevertheless, few works deal with applications of ant colony algorithms to dynamic continuous problems.

Daniel Parrott and Xiaodong Li present in [25] a particle swarm optimization algorithm for multimodal optimization. T. M. Blackwell uses in [4, 5] a technique to solve dynamic problems, which consists in maintaining diversification, by attributing to particles a repulsion charge, in analogy with the electrostatic charges. The PSO method proposed in [25] is a model dedicated to dynamic continuous environments containing several optima.

Genetic algorithms were largely applied to the dynamic landscapes, but mostly in the discrete case. Jürgen Branke (refer to [6], [7]) classifies the dynamic methods found in the literature as follows:

- The reactive methods (refer to [19], [28]), which react to changes (i.e.: by triggering the diversity). The general idea of these methods is to do an external action when a change occurs. The goal of this action is to increase the diversity. In general the reactive methods based on a population loose their diversity when the solution converges to the optimum; this last point causes a problem when the optimum changes. So by increasing the diversity the diversity is regenerated.

- The methods that maintain the diversity (refer to [18], [9], [22], [23]). These methods maintain the diversity in the population, hoping that, when the objective function changes, the distribution of individuals in the search space permits to find quickly the new optimum (see also [16], [26], [15], [1] and [27]).

- The methods that keep in memory “old” optima. These methods keep in memory the evolution of different optima, to use them later. These methods are especially effective when the evolution is periodic (refer to [3], [29] and [2]).

- The methods that use a group of sub-populations distributed on different optima (refer to [31], [24], [8] and [30]), this way the probability to find new optima is increased.

3 Dynamic Hybrid Continuous Interacting Ant Colony algorithm

3.1 CIAC:

A new method introduced by Johann Dréo et al., called CIAC [12], focuses on the principles of communication channels, used in a multi-agent ant colony approach. Thus, a formalization of information exchanges is proposed. Indeed, there are several ways to communicate information between two groups of individuals. From a metaheuristic point of view, there are three characteristics for these communication channels:

1. Perception: the number of individuals concerned by information exchange. Information can be sent, for example, by an individual and received by many others, and vice versa.

2. Memory: the information persistence in the system. Information can remain a certain time in the system or can be just transitory.

3. Integrity: the use of a communication channel can cause modifications. Information can change over time.

Information passing through a communication channel can be any information of interest, such as the value and/or the position of a point in the search space.

CIAC proposes to add to the stigmergic process (which uses the “pheromones”) a direct information exchange, inspired from the heterarchy concept.

3.2 DHCIAC

The DHCIAC algorithm (Dynamic Hybrid Continuous Interacting Ant Colony) [13] is a multi-agent algorithm, based on the exploitation of several communication channels and has two variants, DHCIAC track and DHCIAC find. We use in this work only the DHCIAC track algorithm, that uses the method of the ant colony [10] for global search, and the dynamic simplex method [32] for local search. The ants have two ways of communication, as in the CIAC algorithm. They can deposit pheromone spots in the search space, or use a direct communication channel:
1. The stigmergic channel uses pheromone spots, deposited on the search space, which will be more or less attractive for the artificial ants, according to their concentration and their distance. The characteristics of the stigmergic channel are the following: the range is set at its maximum (every ant has a visibility range), all ants can potentially receive information; there is use of memory, since the spots persist within the search space; finally, information changes over time, since the spots evaporate. The information included in the spots implicitly contains the position of a point and explicitly the value of evolution by the ant having deposited the spot.

2. The direct channel is implemented as a form of exchange of messages between two individuals. An artificial ant has a set of received messages and can send messages to another ant. The range of this channel is set to one, since only one ant receives the message; the memory is implemented in the messages set that ants keep in memory; finally, information (in our case the position and the value of a point) does not change over time.

Each ant chooses between these two channels according to a threshold value. The motivation is a numerical value which changes during the execution. We note that ants perceive only a small region of their environment, called the visible zone.

The first step of the algorithm presented in figure 1 is to randomly put the ants over the search space according to a uniform distribution and to initialize all parameters. Then the pheromonal spots evaporate. At this step, a decision is taken, according to a threshold choice function: the ant chooses to handle one of the two communication channels. Here the two parameters of the stimulus-response function, for the threshold and for the power, are set for the whole population, as each ant has a particular stimulus parameter initialized at the first step according to a normal distribution. If the ant chooses the stigmergic channel, it looks for pheromonal spots in the visible zone. If there is some spots, it moves towards the weighted gravity center of the visible spots, else it goes on the step of motivation decision. On the contrary, if the ant chooses the direct channel, it looks for messages in its message queue. If there is some messages, it moves towards the location indicated by the message and then adds some noise to its new location, else it goes on the step of motivation decision. If there is no spot and no message, the ant takes a decision based on its motivation. The choice is made according to a stimulus-response function. The first choice leads to the direct channel, with message handling.

At this step, another decision is taken according to a stimulus-response function with the same parameter as the previous step, except that the stimulus is \((1 - \text{motivation})\). If the choice is made to handle messages, a message is sent to a random ant and the motivation is increased of a small amount. The second choice is to avoid messages, then the ant goes to the step of random move. If the motivation choice is to handle local search, a dynamic simplex search algorithm \([32]\) is launched with the location of the ant as base point and the radius of the visible zone as initial step length. The local search is limited to a number of evaluations of the objective function.
function. After this local search, the ant looks for visible spots. If there is a spot, it reinforces it and the radius of the visible zone is reduced to the distance to the farthest visible spot. Else if there is no spot, the ant deposits a new one, with a concentration set to the value of the objective function at this ant’s current location. After the handling of the stigmergic channel, the motivation is set to zero.

3.3 Parameter tuning

After a series of preliminary tests on the whole set of test functions described in section 4.2 and in Appendix A, the different parameters of the algorithm were fixed as follows. The dimension of the search space is 2D, and the search interval is [-100,+100]. The number of ants was taken equal to 10, 20, 40, 80 for each function, this parameter is critical when the number of dimensions of the problem increases. The agents must choose between answering to messages from other ants or doing a local search; the probability of this choice was fixed at 0.5, with a standard deviation of 0.2.

The pheromone persistence value $\rho$, if selected high, will trap ants into local optima. The selected value of $\rho$ is 0.9. The threshold under which the pheromone disappears is set to 0.0001. Every simple move of the ants has a value of 0.5. For the dynamic simplex the maximum number of iterations is set to 100, and for DHCIAC 50000. $\sigma$ is the percentage of the size of the search space which will be explored, to determine the standard deviation of the normal distribution of ant ranges. This value allows to determine how the search space will be explored. A “small” value means a bad diversification or exploration, and a “very high” value means an useless dispersion, since ants will be dispersed and search will be random.

The simplex method that we used is the dynamic simplex which uses the reflection coefficient only.

4 Tools used in dynamic optimization

4.1 Performance metrics

The performance measurement of a metaheuristic is much more difficult in the dynamic case than in the static case. Despite the growing interest concerning the evolutionary algorithms in the dynamic case, there is no uniform vision of performance measurement in the dynamic optimization. The majority of the papers dealing with the dynamic case do not describe the test problems or the method used for performance measurement. In [20] the methods are classified as follows:

- A measurement of the difference between the found value of the optimum (the better individual in the population) and the true value of the optimum, just after a change took place in the objective function.
- A measurement of the average of the Euclidean distance to the optimum at each generation.
- The average of the best generations, over several executions of an algorithm on a precise problem.
- Or the difference between the best and the worst generation in a small set of recent generations.

We will use the following performance measurement metric hereafter:

$$\text{Error for value} = \frac{\text{mean}\{ | ROV_{t_j} - BFOV_{t_j} | \}}{}$$

$$\text{Error for position} = \frac{\text{mean}\{ | ROP_{t_j} - BFOP_{t_j} | \}}{}$$

where $ROV_{t_j}$ and $BFOV_{t_j}$ are the true value of the optimum and the best found value of the optimum at time $t_j$ respectively, and $ROP_{t_j}$ and $BFOP_{t_j}$ are the true position of the optimum and the best found position of the optimum at time $t_j$ respectively (with $j \in \{0, \ldots, N\}$). To reduce the fluctuation in the result output due to random initialization, every presented result is the average of 100 executions of the algorithm.
4.2 Benchmark functions

Many test functions were defined in the literature like in [21]. To carry out comparisons between different approaches, it is necessary to consider the same set of problems or functions. Therefore we describe in this section a set of test functions (see on Table 1), which includes several types of dynamic functions [14], from a simple function with a single optimum to functions having a large number of local optima. We point out here that we are dealing with dynamic continuous functions only.

We describe in this section a set of test functions (see on Table 1), which includes several types of dynamic functions [14], from a simple function with a single optimum to functions having a large number of local optima. We point out here that we are dealing with dynamic continuous functions only. Suppose that \( f(x, t) \) is the general form of a dynamic objective function. We can then distinguish two main categories of dynamic optimization functions: the first one concerns a change in the structure itself of the objective function \( f \), and the second concerns the evolution of \( t \), called the “time function”, where \( t \) represents time. A simple example of two dynamic functions, based on traditional static functions, Easom and \( B2 \) respectively, is given below:

\[
f(\vec{x}, T(t)) = -\cos(x_1)\cos(x_2)e^{-(x_1-T(t))^2-(x_2-T(t))^2}
\]

where the position of the optimum is the only element that changes, and:

\[
f(\vec{x}, T(t)) = (x_1 - T(t))^2 + 2 \cdot (x_2 - T(t))^2 + 0.7 - 0.3\cos(3\pi - (x_1 - T(t))) - 0.4\cos(4.\pi - (x_2 - T(t)))
\]

where the whole function changes. The most commonly used variations in dynamic functions are the following: a variation in the objective function and its optimum, a variation in the optimum only, and a variation in the objective function but not its optimum.

We can also mention other more complex variations, like dimensions, constraints, phase (i.e. rotation around an axis) or a complete change (for example, a shift from a convex to a concave form). The time function \( T(t) \) can be linear, periodic (for example \( T(t) = 2 \cdot \cos(t) \)) or non linear.

Eight dynamic test functions are built based on traditional static functions:

- **AbPoP** based on the *Morrison* function: all the structure of \( f \), except the position, changes periodically.
- **AbVP** based on the *Morrison* function: all the structure, except the value, changes periodically.
- **ADL** based on the *Rosenbrock* function: all dimensions change linearly.
- **APoL** based on the *B2* function: the position changes linearly.
- **APhL** based on the *MartinGady* function: the phase changes linearly.
- **AVP** based on the *Shekel* function: the value changes periodically.
- **OPoL** based on the *Easom* function: the position of the optimum changes linearly.
- **OVP** based on the *Morrison* function: the value of the optimum changes periodically.

The first letters mean **A**: for All, **Ab**: for All but, **Po**: for Position, **V**: for value, **Ph**: for Phase. The last letter means **P**: for periodical, **L**: for linear.

The exact definition of each test function is given in Appendix A.

5 Results and discussion

5.1 Tests problems

To test the DHCIAC algorithm described in section 4.2, we used the set of dynamic test functions described in section 4.2 and listed in Appendix A.

5.1.1 Problem 1: AbPoP

The tested function is AbPoP *All but Position Periodic* (Appendix A) based on the static Morrison function. On figure 2 (a) the relative error on the value of the optimum decreases after 25 iterations and is stabilized on the value \(-2.7\). The figure 2 (b) shows the relative error on position of the optimum, that changes periodically over time. We can observe oscillations until the iteration 25, then the value of error is stabilized on a value close to zero. In Table 2 the best error for the value corresponds to 80 agents with a 3.86 standard deviation. And for the position (Tables 3 & 4), the best value corresponds also to 80 agents, with a standard deviation of (3.99, 2.56), best values are in bold character. The first observation concerning the AbPoP function is that DHCIAC could detect the value of the optimum after a small number of iterations. Nevertheless the tracking is less performing. DHCIAC
succeeds to find the position of the optimum after
ants with a standard deviation of 3.49. And for the
until the iteration 10, then it is stabilized on the
the relative error on the position on different di-
ition. The figure shows some oscillations
the position of the optimum, which changes period-
(Figure 3 (a) shows the relative error on
(Appendix A) based on the static Morrison func-
function. The APhL function represents a dis-
keeps an approximately fixed distance to the value
of the optimum.

5.1.2 Problem 2: AbVP

The tested function is AbVP All but Value Periodic
(Appendix A) based on the static Morrison function. The figure 3 (a) shows the relative error on the position of the optimum, which changes periodically over time. The figure shows some oscillations on the error between the values 1 and 3 until the iteration 75, where the error increases to reach the value 13. On figure 3 (b), every curve represents the relative error on the position on different dimensions. The error oscillates between -60 et 100 until the iteration 10, then it is stabilized on the value 0. In Table 2 the best value corresponds to 40 ants with a standard deviation of 3.49. And for the position (Tables 3 & 4) the best value corresponds again to 40 agents, with a standard deviation of (2.54, 2.62). For the test function AbVP, DHCIAC succeeds to find the position of the optimum after a small number of iterations, and succeeds as well to track it at a close distance. Nevertheless, for the value of the optimum, the relative error changes periodically, which means that DHCIAC succeeds in detecting the optimum, but not in tracking it.

5.1.3 Problem 3: APhL

The tested function is APhL All Phase Linear
(Appendix A) based on the static Martin-Gady function. The APhL function represents a dis-
function. The APhL function represents a dis-

Table 1: Test functions ID with corresponding time functions and equations

<table>
<thead>
<tr>
<th>Function ID</th>
<th>T(t)</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbPoP</td>
<td>cos(t)</td>
<td>( \min \sum_{i=1}^{5} \left( -H_i + R_i \left( (x_1 - P_{i,1}(T(t)))^2 + (x_2 - P_{i,2}(T(t)))^2 \right) \right) )</td>
</tr>
<tr>
<td>AbVP</td>
<td>cos(t)</td>
<td>( \min \sum_{i=1}^{5} \left( -H_i + R_i \left( (x_1 - P_{i,1}(T(t)))^2 + (x_2 - P_{i,2}(T(t)))^2 \right) \right) )</td>
</tr>
<tr>
<td>APhL</td>
<td>t</td>
<td>( (X_1 - T(t))^2 + \frac{(X_1 + X_2 - 10)^2}{X_2 - x_2 - T(t)} )</td>
</tr>
<tr>
<td>OVP</td>
<td>8cos(0.5 t)</td>
<td>( \min \sum_{i=1}^{5} \left( -H_i + R_i \left( (x_1 - P_{i,1}(T(t)))^2 + (x_2 - P_{i,2}(T(t)))^2 \right) \right) )</td>
</tr>
<tr>
<td>OPoL</td>
<td>t</td>
<td>( -\cos(x_1) \cdot \cos(x_2) \cdot e^{-((x_1-T(t))^2-(x_2-T(t))^2)} )</td>
</tr>
<tr>
<td>AVP</td>
<td>cos(0.2 t)</td>
<td>( T(t) + \sum_{i=1}^{n} \left( \frac{1}{(x-a_i)^2 + 4c_i} \right) )</td>
</tr>
<tr>
<td>APoL</td>
<td>t</td>
<td>( (x_1 - T(t))^2 + 2 \cdot (x_2 - T(t))^2 + 0.7 \cdot 3 \cdot \cos(3\pi - (x_1 - T(t))) - 0.4 \cdot \cos(4\pi - (x_2 - T(t))) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Agents number</th>
<th>10</th>
<th>20</th>
<th>40</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbPoP</td>
<td>4.21(3.62)</td>
<td>4.15(3.69)</td>
<td>4.33(3.78)</td>
<td>4.09(3.86)</td>
</tr>
<tr>
<td>AbVP</td>
<td>9.99(3.36)</td>
<td>9.79(3.52)</td>
<td>9.73(3.49)</td>
<td>9.89(3.37)</td>
</tr>
<tr>
<td>APhL</td>
<td>4.61(32.09)</td>
<td>2.53(16.51)</td>
<td>1.97(9.14)</td>
<td>2.04(14.14)</td>
</tr>
<tr>
<td>OVP</td>
<td>0.6(4.19)</td>
<td>0.42(4.25)</td>
<td>0.38(4.18)</td>
<td>0.37(4.17)</td>
</tr>
<tr>
<td>OPoL</td>
<td>6.54E-005(0)</td>
<td>1.33E-004(0)</td>
<td>2.23E-004(0.01)</td>
<td>4.85E-004(0.01)</td>
</tr>
<tr>
<td>AVP</td>
<td>10.62(0.14)</td>
<td>10.67(0.19)</td>
<td>10.68(0.18)</td>
<td>10.63(0.12)</td>
</tr>
<tr>
<td>APoL</td>
<td>2.42(2.05)</td>
<td>2.49(2.08)</td>
<td>2.54(2.07)</td>
<td>2.45(2.08)</td>
</tr>
</tbody>
</table>

Table 2: Variation of the value in function of the agents (ants) number.
for the optimum position (Tables 3 & 4), the best error value corresponds again to 40 agents, with standard deviations on $X_1$ and $X_2$ of 52.76 and 31.36 respectively. For the APhL function the convergence towards the optimum position is slow, with large values of standard deviation for the optimum value and position errors. On the other hand the distance between the real and the found optimum is large, which means a dispersion around the optimum. This is due to the discontinuity in the position of the optimum and to the absence of an attraction pool around the optimum.

5.1.4 Problem 4: OVP

The tested function is OVP Optimum Value Periodic (Appendix A) based on the static Morrison function. Figure 5 (a) shows the variation of the real and found optimum values over time. The value of the optimum changes periodically (see Table 1), the algorithm tracks the optimum value until the iteration 110, then the distance between found and real optima increases. Figure 5 (b) represents the error on the position for each dimension. The error oscillates between 0 and 100, then it is stabilized at 0 after 60 iterations. In Table 2 the best value of the error corresponds to 80 agents with a standard deviation of 4.17. And for the optimum position (Tables 3 & 4), the best error value corresponds again to 80 agents, with a standard deviation of (1.59, 1.63). For the test function OVP, DHCIAC succeeds in detecting and tracking periodically the optimum value and positions $X_1$ and $X_2$. The variations at the beginning of the curve are due to the fact that a local optimum of the OVP function changes and becomes a global optimum.

5.1.5 Problem 5: OPoL

The tested function is OPoL Optimum Position Linear (Appendix A) based on the static Easom function. Figure 6 represents the tracking of the optimum position (which changes linearly over time) in function of the ants number (10, 20, 40, 80 ants). The starting point as well as the real optimum variations are indicated. The optimum position moves away directly after the starting point, then gets back and follows linearly the optimum. The convergence is faster with 10 ants, but the tracking is not at a fixed distance. With 80 ants, convergence is slower, but the tracking is closer. In Table 2 the smaller error corresponds to 10 agents with a standard deviation of 0. And for the optimum
Figure 2: Relative errors for AbPoP.

Figure 3: Relative errors for AbVP.
(a) Error on value.

(b) Variation of the position.

Figure 4: Results for APhL.

(a) Value variations.

(b) Error on position.

Figure 5: Results for OVP.

Figure 6: The tracking of the optimum position for OPoL.
position (Tables 3 & 4) the best error value corresponds to 80 agents, with a standard deviation of (15.54, 16.34). For the OPoL function, the test results are promising, and the tracking is done after several oscillations at the beginning of the curve.

5.1.6 Problem 6: AVP
The tested function is AVP All Value Periodic (Appendix A) based on the static Shekel function. On figure 7, the curve shows the error on value for the function AVP. The error oscillates between $-9.5$ and $-11.5$. In Table 2 the best value of the error corresponds to 10 agents with a standard deviation of 0.14. And for the optimum position (Tables 3 & 4) the best error value corresponds to 40 agents, with standard deviations on $X_1$ and $X_2$ of 63.26 and 56.5 respectively. For the test function AVP, DHCIAC does not succeed in tracking the optimum value. Nevertheless the standard deviations are small for optimum value and position.

5.1.7 Problem 7: APoL
The tested function is APoL All Position Linear (Appendix A) based on the static B2 function, the optimum position changes linearly over time. Figure 8 shows the tracking of the optimum position in 3D space. The found value moves away, then gets back at steady intervals. And the distance to the optimum diminishes more and more. In Table 2 the best value of the error corresponds to 10 agents with a standard deviation of 2.05. And for the optimum position (Tables 3 & 4) the best error value corresponds again to 10 agents, with standard deviations on $X_1$ and $X_2$ of 2.96 and 2.88 respectively. The results of DHCIAC for the APoL test function are promising as shown on figure 8, with small values of standard deviation for the optimum value and position.

5.2 Success rates

<table>
<thead>
<tr>
<th>Dynamic function</th>
<th>Position</th>
<th>Value</th>
<th>Success rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbPoP</td>
<td>6+T(t),6</td>
<td>2</td>
<td>51</td>
</tr>
<tr>
<td>AbVP</td>
<td>6,6</td>
<td>2+T(t)</td>
<td>68</td>
</tr>
<tr>
<td>APoL</td>
<td>T(t),T(t)</td>
<td>0</td>
<td>48</td>
</tr>
<tr>
<td>OPoL</td>
<td>T(t),T(t)</td>
<td>0</td>
<td>82</td>
</tr>
</tbody>
</table>

Table 5: Success rates for all the benchmark functions.
The algorithm shows better performances on the APoL function where the optimum position changes linearly, with a success rate of 83%, and for AbPoP, where the structure of the objective function changes, but not the position, with a success rate of 68%. Bad performance is observed for the APhL function where the optimum position presents a discontinuity (see on Appendix A); the success rate is only of 13%.

5.3 Comparison with other algorithms

Despite the growing interest for optimization in dynamic conditions, there exists no common approach on how to compare different algorithms. We mention that almost all results presented in the literature concern discrete problems. We compared our algorithm to the Monte-Carlo method, and to the dynamic simplex method, for all the test functions. Comparisons concern the mean errors on optimum value and position. Our algorithm is also compared to a particle swarm optimization algorithm, on a modified version of the AbPoP function.

<table>
<thead>
<tr>
<th>Function</th>
<th>DHCIAC</th>
<th>Monte-Carlo</th>
<th>Simplex</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbPoP</td>
<td>0.04(3.99)</td>
<td>32.51(40.41)</td>
<td>-4.82(58.45)</td>
</tr>
<tr>
<td>AbVP</td>
<td>-0.21(2.54)</td>
<td>-14.67(33.41)</td>
<td>-4(58.65)</td>
</tr>
<tr>
<td>APhL</td>
<td>-3.29(1.7)</td>
<td>7.55(31.59)</td>
<td>-4.48(7.5)</td>
</tr>
<tr>
<td>OVP</td>
<td>31.49(52.76)</td>
<td>40.08(53.38)</td>
<td>-1.16(98.41)</td>
</tr>
<tr>
<td>OPoL</td>
<td>-92.95(50.1)</td>
<td>35.84(71.38)</td>
<td>264.87(94.57)</td>
</tr>
<tr>
<td>AVP</td>
<td>-18.21(34.14)</td>
<td>1.03(0.81)</td>
<td>-4.65(58.4)</td>
</tr>
<tr>
<td>APoL</td>
<td>-5.8(2.96)</td>
<td>-25.27(18.61)</td>
<td>-6.22(33.28)</td>
</tr>
</tbody>
</table>

Table 6: Comparison related to the optimal value of the objective function.

<table>
<thead>
<tr>
<th>Function</th>
<th>DHCIAC</th>
<th>Monte-Carlo</th>
<th>Simplex</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbPoP</td>
<td>4.09(3.86)</td>
<td>4.38(4.97)</td>
<td>9.52(57.39)</td>
</tr>
<tr>
<td>AbVP</td>
<td>9.73(3.49)</td>
<td>25.35(31.29)</td>
<td>19.23(57.35)</td>
</tr>
<tr>
<td>APhL</td>
<td>0.38(4.18)</td>
<td>6.18(6.70)</td>
<td>9.09(57.38)</td>
</tr>
<tr>
<td>OVP</td>
<td>1.97(9.14)</td>
<td>31.66(53.35)</td>
<td>2.4(13.98)</td>
</tr>
<tr>
<td>OPoL</td>
<td>10.62(0.14)</td>
<td>11.38(0.74)</td>
<td>11.9(13.77)</td>
</tr>
<tr>
<td>AVP</td>
<td>6.54E-005(0)</td>
<td>410.1(1451)</td>
<td>8.74(57.54)</td>
</tr>
<tr>
<td>APoL</td>
<td>2.42(2.05)</td>
<td>30.40(87.46)</td>
<td>8.33(75.65)</td>
</tr>
</tbody>
</table>

Table 7: Comparison related to the position \(X_1\) of the optimum.

In Tables 6, 7 and 8, the three methods DHCIAC, Monte-Carlo and dynamic simplex are compared. The maximal iteration number is set to 5000 for the three methods, and the ant number for DHCIAC is set to 80. For the AbPoP test function, DHCIAC shows better performances than the two competing methods, the errors are in value 4.09(3.86) and in position 0.04(3.99), -0.8(2.56). For the AbVP test function, the ant number for DHCIAC is set to 40 and DHCIAC shows better performances than the other two methods, the errors are in value 9.73(3.49) and in position -0.21(2.54), -0.27(2.62). For the OVP test function, the ant number for DHCIAC is set to 40 and DHCIAC shows better performances than the two other methods, the errors are in value 0.38(4.18) and in position -3.29(1.7), -3.18(1.79). For the APhL test function, the ant number for DHCIAC is set to 40 and DHCIAC shows again better performances, the errors are in value 1.97(9.17) and in position -1.16(98.41), -9.98(13.14). For the AVP test function, the ant number for DHCIAC is set to 40 and DHCIAC shows better performances than the two other methods in the value of the optimum 10.62 (0.14). Monte-Carlo has better performance in position 35.84, 22.88, but with an higher value of standard deviation. DHCIAC presents the smallest standard deviation in position. For the OPoL test function, the ant number for DHCIAC is set to 80 and DHCIAC shows better performances for optimum value than the two other methods 6.54E-005(0). Monte-Carlo has better performances for the optimum position 1.03(0.81), 0.82(0.72). For the APoL test function, the ant number for DHCIAC is set to 10 and DHCIAC shows better performances for optimum value 2.42(2.05) and for optimum position -5.8(2.96), -5.9(2.88).
A comparison with a classical PSO and a variant of PSO is performed. The test function used is based on the static Morrison function with the following time function: \( Y_t = A Y_{i (-1)} (1 - Y_{i (-1)}) \). To be able to compare our method to the PSO methods, we have modified the AbPoP test function, the newly defined function is the following (refer to [21]):

\[
f(x, y) = \max_{i, N} | H_i - R_i \times \sqrt{(X - x_i)^2 + (Y - y_i)^2} |
\]

with \( Y_i \) as a time function, \( N = 3 \), \( H_{base} = 3 \), \( H_{range} = 3 \), \( R_{base} = 2 \), \( R_{range} = 5 \), \( A = 1.2 \).

\[ H_i \in [H_{base}, H_{base} + H_{range}] \]
\[ R_i \in [R_{base}, R_{base} + R_{range}] \]

The test environment is a 2D space within an interval \([-1.0, 1.0]\), the maximal number of iterations is 500, and the results shown in Table 9 are obtained by averaging 50 executions. Agents number (particles and ants) is successively set to 30, 60, 90, 120, 150.

### Table 9: Comparison with two PSO algorithms.

<table>
<thead>
<tr>
<th>Agents number</th>
<th>mean &amp; SD (variant of PSO)</th>
<th>mean &amp; SD (DHCIAC)</th>
<th>mean &amp; SD (classical PSO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>0.1 (0.13)</td>
<td>0.19 (0.32)</td>
<td>0.69 (0.56)</td>
</tr>
<tr>
<td>60</td>
<td>0.07 (0.11)</td>
<td>0.14 (0.29)</td>
<td>0.78 (0.42)</td>
</tr>
<tr>
<td>90</td>
<td>0.06 (0.09)</td>
<td>0.12 (0.22)</td>
<td>0.82 (0.20)</td>
</tr>
<tr>
<td>120</td>
<td>0.05 (0.08)</td>
<td>0.1 (0.13)</td>
<td>0.84 (0.18)</td>
</tr>
<tr>
<td>150</td>
<td>0.04 (0.07)</td>
<td>0.06 (0.09)</td>
<td>0.86 (0.21)</td>
</tr>
</tbody>
</table>

6 Conclusion

The first observation resulting from our experimentation is that DHCIAC could follow the position of the optimum at an almost constant distance, for periodic test functions, for example for the AbPoP function, where the distance to the optimum is almost zero. And when the distance to the optimum increases, the algorithm continues to track the optimum, for example on the AbVP function and the OVP function. This is due to the fact that the distances between optima are constant, and that the dynamic simplex method hybridized with the ant colony algorithm converges rather rapidly to the optimum, for example on the AbVP function and the OVP function. This is due to the fact that the optimum, for example on the AbVP function and the OVP function. This is due to the fact that the optimum is good for the value and the position.

For the sake of the algorithm, the best performances concern functions with value and position variations, and bad performances concern the function with phase variations. In fact, if the random initialization occurs on a plane surface, the dynamic simplex method behaves in a chaotic manner, the successive reflections used by the dynamic simplex do not lead to better values. In the literature, few results exist on continuous dynamic environments. To compare our algorithm to other methods, we have undertaken the same tests on the dynamic simplex method, and the Monte-Carlo method. Our algorithm DHCIAC performs better on the whole set of test functions. A comparison was performed with two particle swarm optimization algorithms. DHCIAC performed slightly worse than the variant of PSO and better than the classical PSO.

### A Test functions for dynamic optimization problems

#### AbVP

**AbVP All but Value Periodic** based on the static Morrison function.

The time function is \( T(t) = \cos(t) \).

\[
D(x_1, x_2, t) = \min_{i} \sum_{i=1}^{5} \left( \left( -H_i + R_i \left( (x_1 - P_{i,1} (T(t)))^2 + (x_2 - P_{i,2})^2 \right) \right) \right)
\]

\( H = \{ 2, 3, 5, 7, 9 \} \)
\( R = \{ 2, 5, 7, 3, 6 \} \)
\( P = \{ 6 + T, 5 + T, 1 + T, 8 + T, 4 + T \} \)

#### AbPoP

**AbPoP All but Position Periodic** based on the static Morrison function.

The time function is \( T(t) = \cos(t) \).

\[
D(x_1, x_2, t) = \min_{i} \sum_{i=1}^{5} \left( \left( -H_i + R_i \left( (x_1 - P_{i,1} (T(t)))^2 + (x_2 - P_{i,2})^2 \right) \right) \right)
\]

\( H = \{ 2 + T, 3 + T, 5 + T, 7 + T, 9 + T \} \)
\( R = \{ 2, 5, 7, 3, 6 \} \)
\( P = \{ 6, 5, 1, 8, 4 \} \)

\( \{ 6, 5, 1, 8, 4 \} \)
Figure 9: Plotting of AbPoP test function.

Figure 10: Plotting of AbVP test function.

Figure 11: Plotting of APhL test function.
**APhL**

APhL All Phase Linear based on the static MartinGady function.
The time function is \( T(t) = t \).
\[
(X_1 - X_2)^2 + \left(\frac{X_1 + X_2 - 10}{3}\right)^2
\]
with \( X_1 = X_{p1} - T(t) \quad X_2 = X_{p2} - T(t) \)
\( X_{p1}, X_{p2} \) are the polar coordinates

**OVP**

OVP Optimum Value Periodic based on the static Morrison function.
The time function is \( T(t) = 8 \cdot \cos(0.5t) \).
\[
D(x_1, x_2, t) = \min \sum_{i=1}^{n} \left(-H_i + R_i \cdot \left((x_1 - P_{1i})^2 + (x_2 - P_{2i})^2\right)\right)
\]
\( H = \{1, 3, 5, 7 + T(t), 9\} \)
\( R = \{1, 1, 1, 1, 1\} \)
\[
P = \begin{cases} 
2.5 & -2.5 \\
-5 & 0 & -4 \\
2.5 & 5 & 0 & -4 
\end{cases}
\]

**OPoL**

OPoL Optimum Position Linear based on the static Easom function.
The time function is \( T(t) = t \).
\[
\text{Easom}(x_1, x_2) = \cos(x_1) \cdot \cos(x_2) \cdot \exp(-(x_1-t)^2-(x_2-t)^2)
\]

**AVP**

AVP All Value Periodic based on the static Shekel function.
The time function is \( T(t) = \cos(0.2t) \).
\[
D(x) = \frac{1}{\sum_{i=1}^{n} (x - a_i)^2} (x - a_i) + c_i
\]
with:
\[
x = (x_1, x_2, x_3, x_4)^T
\]
\[
a_i = (a_{i1}, a_{i2}, a_{i3}, a_{i4})^T
\]
and:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( a_{i1} )</th>
<th>( a_{i2} )</th>
<th>( a_{i3} )</th>
<th>( a_{i4} )</th>
<th>( c_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0.1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>0.2</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>0.4</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>7</td>
<td>3</td>
<td>7</td>
<td>0.4</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>9</td>
<td>2</td>
<td>9</td>
<td>0.6</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>0.3</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>0.7</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>3.6</td>
<td>7</td>
<td>3.6</td>
<td>0.5</td>
</tr>
</tbody>
</table>

**APoL**

APoL All Position Linear based on the static B2 function.
The time function is \( T(t) = t \)
\[
f(\vec{x}, T(t)) = (x_1 - T(t))^2 + 2 \cdot (x_2 - T(t))^2 + 0.7
\]
\[-0.3 \cdot \cos(3\pi - (x_1 - T(t))) - 0.4 \cdot \cos(4\pi - (x_2 - T(t)))
\]

**References**


Figure 12: Plotting of OVP test function.

Figure 13: Plotting of OPoL test function.

Figure 14: Plotting of APoL test function.