A centralised cooperative strategy for continuous optimization: the influence of cooperation in performance and behaviour

Antonio David Masegosa\textsuperscript{a,b}, David A. Pelta\textsuperscript{a,b}, José Luis Verdegay\textsuperscript{a,b}

\textsuperscript{a}Models of Decision and Optimisation Research group, Center for Research on ICT, University of Granada, 18071, Granada, Spain. e-mail: admase@decsai.ugr.es
\textsuperscript{b}Models of Decision and Optimisation Research group, Department of Computer Sciences and Artificial Intelligence, University of Granada, 18071, Granada, Spain. e-mail: \{dpelta, verdegay\}@decsai.ugr.es

Abstract
The necessity of developing high-performance resolution methods for continuous optimisation problems has given rise to the emergence of cooperative strategies which combine different self-contained metaheuristics that exchange information among them. However, the majority of the proposals found in the literature make use of population-based algorithms and/or employ a cooperation scheme with a pipeline or decentralised information flow. In this work we proposed a centralised cooperative strategy where a set of trajectory-based methods are controlled by a rule-driven coordinator. In this context, we also present a new analysis that allows to study the behaviour induced by a determined type of cooperation in the strategy. A comprehensive experimentation has been accomplished over CEC2005 and CEC2008 benchmarks in order to assess the performance of the method with different cooperation schemes. The results show that these cooperation schemes, apart from having a different performance, lead the strategy to distinct exploration and exploitation patterns. In addition, the proposed method presents competitive results with respect to state-of-the-art algorithms for both benchmarks.

Keywords: Continuous optimization, Trajectory-based hybrids, Centralised cooperative strategies, Cooperative behaviour analysis

1. Introduction
Continuous or numerical optimisation is an important research field with many real applications for which different metaheuristics have been proposed. Some of the most relevant methods proposed in the literature are Genetic Algorithms [10, 15], Memetic Algorithms [32, 37, 40], Differential Evolution [39, 42, 46, 57, 61], Particle Swarm Optimisation (PSO) [30, 38, 49, 58], Ant Colony Optimization (ACO) [24, 31] or Tabu Search [20, 34] among others. Although these methods have shown to be effective, their performance usually has a strong dependence on the characteristics of the instance to solve or on their parameters set up. The necessity of developing methods that show a more powerful and robust performance over a wide set of instances have given rise to the emergence of hybrid optimisation algorithms. One of the most successful hybridization alternatives in continuous optimisation has been the combination of different self-contained metaheuristics that cooperate among them. This cooperation can be accomplished by the sequential assembling of different metaheuristics or by establishing a mechanism to exchange...
information among the self-contained methods while they carry out their own search. These hybrids are classified as High-Level Relay (HRH) and Team-work (HTH) following Talbi’s classification [51].

The majority of these methods implement a cooperation scheme based on pipeline, where the output of an algorithm is connected to the input of the next one, or decentralised, in which the exchange of information among the optimisation methods is done directly among them. Another very common characteristic in continuous optimisation hybrids is the use of population-based algorithms (PBAs), that is, hybrids are usually built combining PBAs and trajectory-based algorithms (TBAs) or only PBAs. Virtually no attention has been paid in continuous optimisation to neither centralised cooperation schemes where communication is carried out through a central memory or processor, nor trajectory-based hybrids, and not to mention the combination of both aspects. However, trajectory-based centralised cooperative strategies has been successfully used in other areas as combinatorial optimisation, where they have been applied to problems as the p-median [8], capacitated network design [7], circuit partitioning problem [1], knapsack problem [44] or quadratic assignment [26] among others. Apart from combinatorial optimisation, these strategies can also be found in other fields as Dynamic Optimisation problems [16].

Although the election of the metaheuristics used as optimizers for a centralised cooperative strategy has a noteworthy effect on its performance, the scheme of cooperation also plays a relevant role. Some studies show how different cooperation modes can lead to distinct performances and behaviours [54] for the same set of optimizers. However, few efforts [6, 25, 54] have been undertaken to establish analysis that allow to determine why a determined cooperation scheme shows a better performance or how it alters the search pattern of the strategy. These types of studies may provide a better understanding about cooperative strategies and being very valuable to develop high-performance cooperation schemes.

In this context, the first contribution of this work is a trajectory-based centralised cooperative strategy for continuous optimisation. As far as we know this type of methods has not been applied before. The strategy is based on an approach previously presented in [9, 16, 17, 44] for combinatorial and dynamic optimization. It consists on a set of optimisations agents/solvers that are run in parallel. To control these agents, there exists a coordinator that receives information about their performance and sends orders to them. The coordinator’s behaviour is defined by a control rule base. The cooperative strategy is composed by tabu searches with different parameter instantiation and four control rules for the coordinator, previously presented in [17], are evaluated. The second contribution is a study based on Fitness Landscape Analysis [35] that makes use of a tracking of the local minima visited by the solvers along the whole search. It allows to determine the behaviour induced by a cooperative scheme as well as how it alters the exploration and exploitation patterns of the strategy. The benchmarks used to asses both contributions are the ones proposed for the CEC2005 [50] and CEC2008 [53] competitions.

This paper is structured as follows. Section 2, recalls related work on cooperative strategies for continuous optimisation. The proposed strategy is presented in Section 3 where the control rules evaluated and the tabu search implemented by the solvers are also described. Section 4 is devoted to the experimental framework: description of the benchmarks for CEC2005 and CEC2008 competitions, implementation details and experimental set up. The results obtained are analysed in Section 5. Within this section, a comparison versus state-of-the-art algorithms for both CEC2005 and CEC2008 benchmark is also shown. The new analysis for cooperation schemes is presented and tested in Section 6. Finally, Section 7 gives the conclusions and future work.
2. Related Work

In this section we review some previous relevant work in cooperative strategies for continuous optimisation problems. We will focus on those hybrids where the metaheuristics are self-contained (High-Level hybrids), since they are the closest methods to our proposal. Within this category, the works related to cooperative strategies can be classified in two classes as we pointed out in the introduction.

In first place we have High-Level Relay Hybrids (HRH). A common approach within this class are the so called Memetic Algorithms (MAs) that adds local search operators to Evolutionary Algorithms in order to improve their exploitation capabilities. Many different proposals can be found in the literature. For example, [32] presented a MA that uses crossover-hill climbing methods as local search operator. [43] show a MA that is provided with multiple local searches that are chosen by means of an adaptive mechanism. A probabilistic memetic framework is proposed in [40]. This method balances the local search and the evolutionary operator by estimating the probability of each process to found the global optima during the search. Another interesting approach can be seen in [36] where the MA presented incorporates CMA-ES as local search operator. Some of these authors propose a new version of these MA to deal with large scale continuous optimization problems in [37]. In this case, the local search operator is a variation of Solis West method. Although the Lamarckian model for MAs (both fitness and genetic code of solutions are modified after local optimization) is the most commonly used, Baldwinian models (the local optimization changes the fitness but not the genetic code of the individuals) can also be found [60].

Hybridization with local search methods has been also applied to others PBAs. Some of the most relevant examples are [62], where an Estimation of Distribution Algorithm is coupled with two local searches; [27, 42] in which Differential Evolution is combined with chaotic or SPX local search operator; or [5] where the performance of the PSO method is improved by applying a local search process using Extremal Optimization from time to time. Another interesting HRH hybrid for continuous optimisation has been proposed in [47], where a PSO algorithm is combined with Ant Colony Optimisation to intensify the search in every particle after each iteration. Tseng et al. presented in [55] an HRH hybrid composed only by trajectory methods. This strategy counts with three local searches that are used or not depending on their performance.

The main difference between the methods above and our proposal is on how metaheuristics are applied. While our cooperative strategy performs the search by running the optimizers concurrently, these algorithms are applied sequentially as a pipeline where the output of a method is “plugged” to the input of the next one.

The other class of algorithms related to our proposal are High-Level Team Hybrids (HTH). One of the most known approaches of this category are the island model EAs, also known as coarse grained or multi-population. These methods, instead of a unique population of solutions, have several sub-populations that co-evolve in parallel carrying out migrations of individuals among them when a determined condition is fulfilled. The island model has been proposed for EAs as GAs [2, 28, 29], DE [11, 59] or Line-up competition algorithms [48], although this idea have been also taken by other methods as PSO [41, 63]. Peng et al. presented a similar approach in [45]. In this case, different population-based methods are combined in an algorithm portfolio that exchanges solutions among its components.

Hierarchical distributed genetic algorithms [23] are other type of HTH hybrids that have been applied to continuous optimisation. In these algorithms, there exists an island model at a top level which makes cooperate different multi-population GAs in a lower level. The cooperative strategy
proposed in this paper mainly differs from the former methods in the cooperation scheme since it is centralised, that is, the information exchange is done through a central processor. However, the former algorithms implement a decentralised scheme where the information flows directly between algorithms.

Hierarchically organised evolutionary strategies (HOES) [3, 22] are HTH hybrids that can be considered as centralised cooperative strategies. The structure of these methods is split into two levels. The first one is composed by a set of EAs with different parameter set up whereas the second level monitors these EAs to adjust their parameters according to the best performing values. Another centralised cooperative strategy was proposed in [56]. This proposal consist on a self-adaptive multi-method approach called MultiAlgorithm Genetically Adaptive Method for Single Objective Optimization (AMALGAM-SO). The approach combines different PBAs methods that works over the same population and automatically adjust the number of offsprings these algorithms are allowed to contribute during each generation.

Despite of having a centralised cooperation scheme, the purpose of these strategies is different from our proposal since they are mainly designed for PBAs and have an ad-hoc information exchange scheme. In our case, the optimizers are trajectory-based and the information exchange scheme, mainly defined by the coordinator, has a more general purpose.

3. General scheme of the strategy

The centralised cooperative strategy proposed in this work is based on a scheme previously presented in [9, 16, 44]. This strategy consists on a set of solvers/threads, where each solver can implement the same or a different resolution method for the problem at hand. The coordinator processes the information received from the solvers and produces subsequent adjustments of their behaviour by sending “orders”. To achieve this exchange of data, a blackboard model is used [12]. Concretely, two blackboards are available, one for the solvers→coordinator flow (S/C blackboard), in which the information exchanged are reports about solvers’ performance, and another for the coordinator→solvers flow (C/S blackboard), used for the order sending.

After an initialization stage, the solvers are executed asynchronously while sending and receiving information. The coordinator checks through the S/C blackboard which solver provided new information and decides whether its behaviour needs to be adapted using a rule base. In this case, it will determine which is the proper directive and will send it to the solver through the C/S blackboard. As for the solvers, its working it is also very simple: once execution has begun, performance reports are sent and adaptation orders from the coordinator are received alternatively.

Every cooperative strategy of this kind should define an information management strategy providing: type and content of the information exchanged and how such information is processed. These details will be described in the next section.

3.1. The information management strategy

As we have just seen, the information flow in our strategy can be divided in the three following stages: 1) performance report sending from solvers to coordinator, 2) information processing and storing and 3) adaptation directives sent from coordinator to solvers. Before stating the description of each stage it is important to point out that the solvers belong to the class of local-search procedures.

In the first stage, the mentioned performance reports contain the next items:
• solver identification
• time stamp \( t \)
• current solution of the solver at time \( t \) (\( s' \))
• best solution reached until time \( t \) by this solver (\( s_{best} \))
• list with the local minima found by the method since the last report.

The list of local minima consists on an array of solutions sorted according to the moment in which the solution was found. In order to measure the improvement rate of the solvers, the coordinator keeps, for each of them, two arrays, \( T \) and \( FCS \), where it stores the last values of \( t \) and \( f(s') \) reported by the solver, respectively, being \( f \) the objective function. The length of \( T \) and \( FCS \) is given by \( l_{IR} \).

In the next stage (information processing), it calculates the corresponding improvement rate \( \Delta \) as the opposite of the slope of the regression line that has the current solution’s fitness and the time as the dependent and independent variables, respectively. Concretely, it is calculated by the next equation:

\[
\Delta = -\frac{\text{cov}(T, FCS)}{\sigma^2_T}
\]  

where \( \text{cov}(\cdot) \) is the covariance function and \( \sigma^2(\cdot) \) is the variance. The values \( \Delta \) and \( f(s') \) are then stored in two fixed length ordered memories, \( M_{IR} \) and \( M_{C} \), respectively. The list of local minima contained in the report is processed by the coordinator to keep a history of all local optima found by the solvers. This history is stored in a data structure called Visited Region List (VRL). The local minima are considered as regions limited by a hypersphere with radius \( \rho \) and centre in the solution stored in that list. Each entry of the VRL also maintains a register \( \phi \) with the frequency of visits of the corresponding local optimum/region by any search thread.

The next step within the information processing stage makes use of the former information to detect bad behaviour patterns in the solvers, as well as to determine the proper action to correct it. To achieve this task, the coordinator employs a set of control rules that establish when a solver is showing a poor performance and the corresponding adaptation order.

3.2. Control rules

The control rules have the next template:

\[
\text{IF condition THEN action}
\]

where the action part of the rule takes the form

\textbf{action}: send a new solution to the solver \( i \)

Such solution is used by the solver, as a restart point of its search. This is the behaviour implied by the action defined here. In this way, as the strategy progresses, the solver can be placed in promising regions of the search space, non-explored zones, etc.

In this work we will evaluate four different control rules that share the same condition part but they perform different actions. The chosen condition was successfully used in other works as \([9, 44]\), and was designed using fuzzy sets.

The next subsections are devoted to describe this fuzzy condition as well as the evaluated actions.
3.2.1. Fuzzy condition

This fuzzy condition was designed using expert knowledge and its objective is to detect if a solver is "working bad". To do this, the solver's current solution quality and its stagnation with respect to what the coordinator has recently observed in the others optimisers is measured. The definition of this condition is the following:

(quality of the current solution reported by solver_i is low_Q) AND
(improvement rate of solver_i is low_IR)

The labels low_Q and low_IR are defined as fuzzy sets over the memories M_C and M_IR, respectively, and their membership functions \( \mu(x) \) are shown in Figure 1. The variable \( x \) corresponds to the relative position (resembling the notion of percentile rank) of a value (an improvement rate or a cost) in the samples stored in the respective memories (M_C and M_IR). The parameters are fixed to \( a = 60 \) and \( b = 100 \) for the set containing the costs, and \( a = 0 \) and \( b = 40 \) for the set containing the improvements. This way of measuring the quality of the improvement rates and the solution is independent of the problem, instance or scale.

3.2.2. Actions

The actions assessed in this work combine the information available in the coordinator in different manners to generate solutions that lead the strategy to different exploitation/exploration balances. Their descriptions are given below:

- **Best solution (BSC)**

  We can consider this action as the most naive. It consists on sending the best solution ever seen by the coordinator to the solver. In this way, we concentrate the threads around promising regions of the search space. Before being sent, the solution is slightly altered by means of a mutation operator.

- **Approaching (AC)**

  This action tries to bring solvers with bad performance near to the best optimiser, placing it in an intermediate point between the best solution of both solvers. This point is generated applying a crossover operator to these solutions.

- **Reactive (RC)**

  This consequent for the coordinator’s rule was previously presented in [33]. When executed, the coordinator send, to the specific solver, the best global solution perturbed by a certain degree \( \phi \) given by the next formula:
\[ \phi = \begin{cases} 0, & \text{if } \phi - \lambda_{\text{reaction}} \leq 0 \\ \phi - \lambda_{\text{reaction}}, & \text{if } \phi - \lambda_{\text{reaction}} > 0 \text{ and } \phi - \lambda_{\text{reaction}} < \phi_{\text{max}} \\ \phi_{\text{max}}, & \text{if } \phi - \lambda_{\text{reaction}} \geq \phi_{\text{max}} \end{cases} \]

where \( \phi \) is the frequency of visits the region of the last minimum found by the solver, \( \lambda_{\text{reaction}} \) is a threshold which establishes when a higher mutation than the basic one (\( \phi=0 \)) is applied and \( \phi_{\text{max}} \) is the maximum perturbation degree.

- **Visited Region List (VRLC)**

  This action makes also use of the VRL, although here this list is utilised to reallocate the solver in a point outside of the previously visited regions. The procedure followed to generate these points has been taken from [21] and the steps that it does are the following:

  1. Generate a new solution \( x \) randomly.
  2. Compute the quantities:
     \[ d_i = \frac{\|x - m_i\|}{1 + \Psi(\phi_i)}, \quad i = 1, \ldots, M. \]
  3. If \( d_i/\rho \geq 1, \forall i \in [1 \ldots M] \), accept \( x \). Otherwise, return to Step 1.

  In the equations above, \( m_i \) is the \( i \)-th local minimum of the VRL, \( M \) is the size of the VRL and \( \rho \) is the radius of the regions. \( \Psi(\phi) \) is a function whose objective is to avoid the generation of solutions in those more frequently visited regions. Its definition is:

  \[ \Psi(\phi) = \gamma(1 - e^{\gamma(\phi - 1)}) \] (2)

where \( \gamma \in (0, 1] \) is a given constant and \( \phi \) the frequency of visits of the corresponding region. In this way, a point close to more frequently visited regions (with a high value of \( \phi \)) is hardly accepted because of its higher value of \( \Psi \). To avoid infinitely cycling in this process, it is terminated after a predetermined number of iterations.

The first two actions lead to a bigger exploitation balance whereas the others two are more oriented to exploration. We can also check that AC and RC, apart from global information (coordinator’s best solution, regions visited), take into account local information from the solver (solver’s best solution, last local minima found) to determine its new position.

### 3.3. Continuous Tabu Search description

The method used as solver is a tabu search algorithm that can be consider as a hybridization of two metaheuristics previously presented in the literature, DOPE [13] and DTS [21]. From the first one, we have taken the variable move step as well as some parts of its tabu neighbourhood exploration method. As for the second one, its diversification method has been introduced. Before seeing the algorithm in a comprehensive manner, we are going to describe the main components of this method:

#### 3.3.1. Variable move step

An important issue to determine in continuous optimisation when we search for better solutions around a specific point is the size of the movement. A small step size can lead to an important waste of objective function evaluations, whereas a big movement length makes difficult to find solutions with enough accuracy. For this reason, it can be interesting to use a big step size at the beginning of the search to do a better exploration of the search space, and then reducing this size to obtain a better accuracy of the solution. In this way, in our algorithm we start
with a step size of length $\delta_{init}$ and every time the exploration of the neighbourhood of the current solution does not lead to better position, this size is halved. When two consecutive steps get to improve the current solution, the step size is multiplied by two. The length of the movement is delimited by the interval $[\text{res}, \delta_{init}]$, being res a parameter which determines the precision of the algorithm.

### 3.3.2. Mechanisms to escape from local minima

When the algorithm has reduced its step size to a predefined minimal resolution res and none of the neighbours surrounding the current solution has a lower cost value we can establish that the algorithm has reached a local minimum. In order to escape from them, this tabu search uses two mechanisms.

In first place, after finding the minimum, the algorithm continues from that point, reinitialising the size step, and allowing the acceptance of maxNonImp non-improving movements. To avoid falling again in the last minimum, the algorithm keeps a tabu list with the reverse of the movements done in the last tenure iterations.

The other mechanism is triggered when the method fall in a previously visited minimum. The algorithm keep a list MIN with all the local minima $m_i$ found until the current moment. Each local minimum has associated a region of the search space consisting on a hypersphere with radius LRR and centre in $m_i$. An example of this diversification procedure can be seen in Figure 2. Given a point $x$ that lies in a local minimum region, this mechanism performs the following steps:

1. We compute the centroid $\overline{m}$ of the local minimum regions:

   $$\overline{m} = \frac{1}{|M|} \sum m_i$$

2. We construct search directions parallel to the coordinate axes and pointing towards the direction $x - \overline{m}$, i.e, the neighbourhood search directions are determined as

   $$d_j = \text{sign}(x_j - \overline{m}_j) \cdot e_j, \quad j = 1, \ldots, n \quad (3)$$

   where $e_j \in \mathbb{R}^n$ is the $j$-th unit vector in $\mathbb{R}^n$ and $n$ corresponds the dimension of the problem.

3. $n$ neighbours points $p_j$ are generated along these search directions with a random step size $\alpha_j$, that is,

   $$p_j = x + \alpha \cdot d_j, \quad j = 1, \ldots, n \quad (4)$$

Then the best $p_j$ is chosen as the new current solution.
Algorithm 1 Neighbourhood Exploration

parameters: $x$, $\delta$, # of no improvements accepted

$x_{\text{new}} \leftarrow$ OptimisticSearch($x$, last movement, $\delta$)

if NoImprovement($x_{\text{new}}$, $x$) then
    $x_{\text{new}} \leftarrow$ ADD($x$, $\delta$)
    if NoImprovement($x_{\text{new}}$, $x$) then
        $x_{\text{new}} \leftarrow$ TabuExploration($x$, $\delta$)
    end if
end if

return $x_{\text{new}}$

3.3.3. Neighbourhood exploration

One of the main parts of this algorithm is the way in which it explores the neighbourhood. Concretely, this method considers three different exploration modes that are tested in a specific order. If a determined mode does not lead to a successful movement, then the next one is tried. The three exploration forms are described below:

1. **Optimistic search**: In first place our algorithm tries a step in the last good direction, that is, the direction of movement chosen in the previous neighbourhood exploration, since we can hope that this trajectory will be still good in the next iteration.

2. **Approximate descent direction (ADD)**: If the last procedure does not lead to an improvement, then the heuristic attempt to find a good descent direction using the ADD method [19].

3. **Tabu exploration**: In the case that the two former movements do not succeed, the algorithm carries out a more exhaustive search generating $2n$ neighbours. Concretely, for each unit vector $e_j \in \mathbb{R}^n$, two solutions are taken in this direction, one in positive sense and one in negative. Then, the best non-tabu move or the best tabu move (if it fulfills the aspiration level) is taken as new current solution of the method.

For a better understanding of the neighbourhood exploration, its pseudocode is given in Algorithm 1.

3.3.4. Main algorithm

With the aim of a better understanding of the general working of this continuous tabu search, its pseudocode is provided in Algorithm 2.

4. Experimental framework

Different experiments have been performed in order to assess the performance of the trajectory-based centralised cooperative strategy for continuous optimisation problems. In this section we describe the most important aspects of the experimentation carried out as the test suites, the implementation details and the experimental set up.
Algorithm 2 Continuous Tabu Search pseudocode

procedure ContinuousTabuSearch
\[ \delta \leftarrow \delta_{init} \]
\[ x \leftarrow \text{GenerateInitialSolution()} \]
\[ x_{best} \leftarrow x \]
\[ f_{best} \leftarrow f(x_{best}) \]
\[ \text{consAdvances} \leftarrow 0 \]

while not stopping condition do
  if \( \delta < \text{res} \) then
    \[ \delta \leftarrow \delta_{init} \]
  
  if IsNearLocalMinimum(\( x_{best} \)) then
    \[ x \leftarrow \text{Diversification}(x_{best}) \]
    \[ x_{best} \leftarrow x \]
    \[ f_{best} \leftarrow f(x_{best}) \]
    \[ \text{consAdvances} \leftarrow 0 \]
  else
    acceptNoImp \leftarrow \text{maxNoImp}
    AddLocalMinimum(\( x_{best} \))
    \[ f_{best} \leftarrow \infty \]
    \[ \text{consAdvances} \leftarrow 0 \]
  end if

  \[ x \leftarrow \text{ExploreNeighbourhood}(x,\text{acceptNoImp},\delta) \]

  if \( f(x) < f_{best} \) then
    \[ x_{best} \leftarrow x \]
    \[ \text{consAdvances} \leftarrow \text{consAdvances} + 1 \]
  if \( \text{consAdvances} = 2 \) then
    \[ \delta \leftarrow \text{IncreaseStep}(\delta) \]
    \[ \text{consAdvances} \leftarrow 0 \]
  end if
  else
    \[ \delta \leftarrow \text{ReduceStep}(\delta) \]
    \[ \text{acceptNoImp} \leftarrow \text{acceptNoImp} - 1 \]
  end if

end while

4.1. Test functions

Our experimentation was focused on unconstrained non-linear minimisation problems:

\[
\min_{x \in \mathbb{R}^n} f(x)
\]  

being \( f \) a real-valued function defined in \( \mathbb{R}^n \). The solutions are chosen from a range \([L, U] = \{x \in \mathbb{R}^n : l_i \leq x_i \leq u_i, i = 1, \ldots, n\}\). Given that the functions, we will work with, have the same range for all the dimension, then \([L, U] = \{x \in \mathbb{R}^n : l \leq x_i \leq u, i = 1, \ldots, n\}\). The dimension range will be named as \( \varepsilon = u - l \).

Two benchmarks of continuous global optimization functions were considered: the test beds for the competitions proposed in 2005 and 2008 IEEE Congress on Evolutionary Computation (CEC2005 [50] and CEC2008[53]). Their descriptions are given in the following subsections.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>δinit</td>
<td>initial step size</td>
<td>ε/2, ε/3, ε/5</td>
</tr>
<tr>
<td>εres</td>
<td>minimal step size</td>
<td>0.001</td>
</tr>
<tr>
<td>LRR</td>
<td>radius for the local minimum regions</td>
<td>0.01 · ε</td>
</tr>
<tr>
<td>maxNoImp</td>
<td>maximum number of non improvement movements</td>
<td>3</td>
</tr>
<tr>
<td>tenure</td>
<td>length of the tabu list for movements</td>
<td>1</td>
</tr>
<tr>
<td>α</td>
<td>step size for diversification procedure</td>
<td>(0.1 + ω) · ε</td>
</tr>
</tbody>
</table>

Table 1: Parameter setting for the continuous tabu search. The values given to the parameter δinit in the three solvers are also displayed. (ω is a random value uniformly distributed in [0,1])

4.1.1. Benchmark from CEC2005 competition

This test bed is composed by 25 functions designed for the special session on real parameter optimisation of this conference and they can be classified in four groups:

- **Unimodal functions** (f1 – f5)
- **Multimodal basic functions** (f6 – f12)
- **Multimodal expanded functions** (f13 – f14)
- **Multimodal hybrid composition functions** (f15 – f25)

A more detailed description of these functions as well as information about the source code in different programming languages can be found in [50]. All the test functions have been displaced to ensure that the optima can never be located in the centre of the search space. Two of the functions have their optimum out of the range of the initialisation.

4.1.2. Benchmark from CEC2008 competition

This test suite was proposed for the special session on large scale global optimization [53] of CEC2008 and has a total of 7 functions. The functions that compose this benchmark are the followings:

- **Unimodal functions** (f1 – f2)
- **Multimodal functions** (f3 – f5)

As in the former test suite, the functions are displaced in order to avoid the location of the global optimum in the center of the search space. More information about this benchmark can be found in [53].

4.2. Implementation details

When implementing this type of cooperative strategies, one can resort to parallel schemes if time is important, or one can simulate the parallelism in a one-processor computer. This is the strategy taken here and the procedure is extremely simple. We construct an array of solvers and we run them using a round-robin schema. To simulate the asynchronous behaviour, solvers are executed during certain number of evaluations of the objective function. This amount of evaluations is a random number in the interval $[2 \cdot \text{max Evaluations} \cdot \frac{1}{1000}, 3 \cdot \text{max Evaluations} \cdot \frac{1}{1000}]$. After the execution,
the communication with the coordinator takes place. These steps are repeated until the stopping condition is fulfilled.

The cooperative strategy is composed by 3 solvers which implements the method described in Section 3.3. They differ in terms of the initial step length ($\delta_{init}$) as well as the starting solution. The parameter setting for the continuous tabu search was done according to the recommendations given in [13, 21] and is displayed in Table 1. Regarding to the coordinator set up, the length of the data memories $M_{costs}$ and $M_{IR}$ was fixed to 12 and the size of the samples FCS and T ($l_{IR}$) to 10. The radius for the visited regions $\rho$ was set to $0.15\varepsilon$.

Focusing on the implementation details of the four actions or consequents studied, the mutation operator used in BSC to modify the best global solution consists on selecting randomly a direction following a uniform distribution and then, take the point at a distance $r$ from this solution. In this case, $r$ is set to $0.1\varepsilon$.

The intermediate points between two solutions for AC consequent are generated by an average crossover operator: given two solutions $x_1 = \{x_{11}, \ldots, x_{1n}\}$ and $x_2 = \{x_{21}, \ldots, x_{2n}\}$, each component of the resulting solution $z$ is calculated as $z_i = \frac{x_{1i} + x_{2i}}{2}$, $i = 1, \ldots, n$.

For RC, the modifications applied to the best solution are carried out by means of the same operator used in BSC. The different degrees of perturbation are obtained by varying the parameter $r$. Concretely, $r$ is calculated as $(0.1 + 0.05\phi)e$, being $\phi_{max} = 3$. In the VRLC implementation, the parameter $\gamma$ was set to $0.25$.

4.3. Experimental setup

In order to compare the results obtained by our cooperative strategy with other algorithms from the state-of-the-art, we have followed the guidelines given in both CEC2005 and CEC2008 competitions. The main points are the followings:

- Each algorithm was executed 25 times for each test function. We computed the average of error obtained at the end of each run, where the error was calculated as $f(x) - f(x^*)$, being $x^*$ the optimum of the function.

- In CEC2005 benchmark we considered the dimensions 10, 30 and 50, whereas for CEC2008 benchmark we worked with dimension 1000.

- The stopping conditions were set to $10000 \times \text{dimension}$ and $5000 \times \text{dimension}$ objective function evaluations in CEC2005 and CEC2008 test suites, respectively.

In order to compare the different studied methods and to check if there exists significant difference in performance among them we made use of Mann-Whitney’s U non-parametric test for pairwise comparisons at a significance level of $\alpha = 0.05$.

5. Results Analysis

This section is devoted to the analysis of the results obtained in the different experimentations carried out. Concretely, our objectives are:

- Evaluate the performance of the trajectory-based centralised cooperative strategy with the different actions seen in Section 3.2.

- Compare the performance of the method presented in this work against state-of-the-art algorithms for CEC2005 and CEC2008.
Table 2: Comparison of the percentage of functions in which an algorithm (row) performs significantly worse/better than another one (column), for CEC2005 benchmark with dimensions={10, 30, 50} and CEC2008 benchmark with dimension 1000 (Mann-Whitney’s U non-parametric test \( \alpha < 0.05 \))

5.1. Analysing the performance of the coordination modes

This study is oriented to analyse the performance shown by the four actions or solver reallocation schemes when they are incorporated into the strategy. In order to have a baseline case that allow us to have a reference of the effectiveness of these actions we also show the performance obtained by the independent version of our strategy, that is, the three solvers do not exchange information among them. In this way, we will compare five methods: BSC, AC, RC, VRLC and I corresponding to the cooperative strategy working with each of the four studied actions besides the independent strategy.

The results (best solution, mean error and standard deviation) obtained by each of these methods are shown in supplementary Appendix A. To make easier the comparison among the methods, we measured the number of functions in which a determined algorithm has a significantly different performance with respect to other one by the Mann-Whitney’s U non-parametric test. In Table 2, a comparison among each possible pair of algorithms is shown. Given a pair A-B (row-column) of methods, the corresponding cell shows the percentage of functions in which A is significantly worse/better than B in terms of the mean ranking returned by this non-parametric test. The last block of rows in this table displays the mean percentage over all test suites, that is, over 82 functions (75 from CEC2005 (dimensions 10, 30 and 50) + 7 from CEC2008). Last column shows the ranking of the methods for each benchmark and dimension, as well as the global one. The methods are ordered (from up to down) according to the balance of the percentages of functions with significantly better (% better) and worse (% worse) performance. Concretely, in this ranking, for every pair of methods \((A, B)\) with positions \(p_A\) and \(p_B\), the next conditions are fulfilled:

\[
p_A < p_B \Rightarrow \%_{\text{better}}(A, B) > \%_{\text{worse}}(A, B)
\] (6)
\[ P_A = P_B \implies \%_{\text{better}}^{(A,B)} = \%_{\text{worse}}^{(A,B)} \]  

Looking to the global ranking, we can see that I is the worst method, being improved significantly in a percentage nearly or higher to/than 40 whereas the opposite percentage is lower or equal to 13. This is interesting because it shows that any of the different coordination modes can be beneficial for the strategy’s behaviour. Among these coordination modes, AC obtains the best performance with a positive difference of functions with better/worse behaviour that goes from the 15% to 32%. The second position in this ranking is for RC followed by BSC, although in this case the distance between both percentages is low (22%-13%). VRLC is the worst action with an important difference in performance, especially with respect to AC and RC. A striking aspect of these results is the fact that the two best reallocation schemes are those that use local information from the solvers apart from the global one.

Analysing the CEC2005 benchmark separately, we should highlight the increase of the difference in performance between I and the cooperative methods as the dimension is enlarged, in virtually all cases. Nevertheless, the dimension increment hardly modifies the ranking of the algorithms which is quite similar to the global one. There are only two remarkable differences for dimensions 10 (VRLC improves BSC) and 30 (AC and RC perform significantly worse/better in the same number of functions). Roughly, the results in CEC2008 benchmark follow in the same vein. I is the worst algorithm, being improved for all cooperative strategies, specially by AC and RC that show a performance significantly better in the 7 functions of this test suite. VRLC has a poorer behaviour than in the former benchmark, improving I in a significant way only in a 29 percent of the functions and being outperformed by the other coordination modes. AC is also the best method whereas RC and BSC have a complementary performance, that is, both of them improve and are improved in almost half of the functions.

Another interesting point of these results is the high percentage of functions in which some pairs of cooperative strategies show a significantly different performance. For example, the hypothesis of equality can be rejected in a 100% and a 72% of the functions for AC and BSC in CEC2008 and CEC2005 with dimension 50, respectively. A value similar to this last one can be found in CEC2005 with dimensions 30 and 50 for RC and AC, although in this case, the better/worse percentages are more balanced. This suggests that different ways of cooperations leads the strategy to behave in a distinct way and therefore, to show a better or worse performance depending on the characteristics of the function at hand. This fact will be analysed in Section 6.

### 5.2. Cooperative strategies vs. state of the art methods

In order to assess the performance of this centralised trajectory-based cooperative strategy, we compared the independent strategy and the best performing coordination mode (AC) versus five state-of-the-art methods for CEC2005 benchmark and versus the algorithms that participated in the CEC2008 competition. The independent method has been considered in order to have a reference about the performance of the solver implemented by the cooperative strategy.

Starting with CEC2005 test suite, the five methods chosen for the comparison are the next ones:

- G-CMA-ES [4], an improved version of CMA-ES with restart and adapting population size that won the CEC2005 competition [18].

---

1The source codes of CLPSO, JADE and GL-25 were downloaded from http://dces.essex.ac.uk/staff/qzhang/
Table 3: Comparison of I and AC versus G-CMA-ES (G-CMA in the table), MA-LSCh-CMA (MA-CMA in the table), CLPSO, JADE and GL-25 in CEC2005 benchmark.

<table>
<thead>
<tr>
<th>Dim</th>
<th>Method</th>
<th>inst worse</th>
<th>insts better</th>
<th>draws</th>
<th>R</th>
<th>R+</th>
<th>Sig?</th>
<th>insts worse</th>
<th>insts better</th>
<th>draws</th>
<th>R</th>
<th>R+</th>
<th>Sig?</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>G-CMA</td>
<td>4</td>
<td>17</td>
<td>0</td>
<td>16</td>
<td>194</td>
<td>Yes</td>
<td>10</td>
<td>0</td>
<td>132</td>
<td>78</td>
<td>No</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MA-CMA</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>132</td>
<td>78</td>
<td>No</td>
<td>10</td>
<td>0</td>
<td>132</td>
<td>78</td>
<td>No</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>CLPSO</td>
<td>8</td>
<td>12</td>
<td>0</td>
<td>93</td>
<td>117</td>
<td>No</td>
<td>10</td>
<td>0</td>
<td>111</td>
<td>79</td>
<td>No</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>JADE</td>
<td>7</td>
<td>13</td>
<td>0</td>
<td>76</td>
<td>134</td>
<td>No</td>
<td>10</td>
<td>0</td>
<td>103</td>
<td>87</td>
<td>No</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>GL-25</td>
<td>12</td>
<td>8</td>
<td>0</td>
<td>134</td>
<td>76</td>
<td>No</td>
<td>15</td>
<td>5</td>
<td>172</td>
<td>38</td>
<td>Yes</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>G-CMA</td>
<td>6</td>
<td>14</td>
<td>0</td>
<td>66</td>
<td>144</td>
<td>No</td>
<td>5</td>
<td>15</td>
<td>0</td>
<td>38</td>
<td>152</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>MA-CMA</td>
<td>5</td>
<td>14</td>
<td>1</td>
<td>29</td>
<td>161</td>
<td>Yes</td>
<td>5</td>
<td>15</td>
<td>0</td>
<td>20</td>
<td>190</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CLPSO</td>
<td>9</td>
<td>10</td>
<td>1</td>
<td>96</td>
<td>94</td>
<td>No</td>
<td>10</td>
<td>9</td>
<td>1</td>
<td>91</td>
<td>99</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>JADE</td>
<td>6</td>
<td>13</td>
<td>1</td>
<td>63</td>
<td>127</td>
<td>No</td>
<td>6</td>
<td>13</td>
<td>1</td>
<td>60</td>
<td>130</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>GL-25</td>
<td>9</td>
<td>10</td>
<td>1</td>
<td>87</td>
<td>103</td>
<td>No</td>
<td>10</td>
<td>9</td>
<td>1</td>
<td>79</td>
<td>111</td>
<td>No</td>
</tr>
<tr>
<td>50</td>
<td>G-CMA</td>
<td>6</td>
<td>14</td>
<td>0</td>
<td>83</td>
<td>127</td>
<td>No</td>
<td>7</td>
<td>13</td>
<td>0</td>
<td>83</td>
<td>127</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>MA-CMA</td>
<td>8</td>
<td>12</td>
<td>0</td>
<td>54</td>
<td>156</td>
<td>No</td>
<td>9</td>
<td>11</td>
<td>0</td>
<td>54</td>
<td>156</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CLPSO</td>
<td>8</td>
<td>12</td>
<td>0</td>
<td>100</td>
<td>110</td>
<td>No</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>100</td>
<td>110</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>JADE</td>
<td>6</td>
<td>14</td>
<td>0</td>
<td>67</td>
<td>123</td>
<td>No</td>
<td>7</td>
<td>12</td>
<td>1</td>
<td>67</td>
<td>123</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>GL-25</td>
<td>8</td>
<td>12</td>
<td>0</td>
<td>77</td>
<td>133</td>
<td>No</td>
<td>7</td>
<td>13</td>
<td>0</td>
<td>79</td>
<td>131</td>
<td>No</td>
</tr>
</tbody>
</table>

- MA-LSCh-CMA [36], a MA that uses CMA-ES as local search operator and whose main novelty is to allow to this local search operator to start from the final configuration of a previous invocation.
- CLPSO [30], a PSO algorithm that updates the particle’s velocity using the particles’ historical best information. This provides a better diversity and prevents premature convergence.
- JADE [61], a DE method that uses DE/current-to-pbest as mutation strategy with optional external archive and that incorporates a self-tuning mechanism to adapt control parameters.
- GL-25 [15], an hybridization of a global and a local real coded genetic algorithm. This local and global behaviour is obtained by adjusting the parameters of its parent-centric crossover operator.

For this comparison we focused on the 20 multimodal functions of this benchmark (f6-f25) since they are the hardest ones. To assess statistically the differences between these state-of-the-art algorithms and the two versions of our cooperative strategy the Wilcoxon’s paired non-parametric test at significance level of 0.05 was applied following the guidelines given in [14]. We took as samples the mean error over the 25 runs obtained by the method in each function. These results are shown in Table 3. This table displays for each state-of-the-art algorithm the number of functions in which their mean error is worse (insts worse), better (insts better) or equal (draws) than/to the one obtained by I or AC strategies(column), as well as the rank sum returned by the Wilcoxon’s paired non-parametric test (the higher the better) for the state-of-the-art algorithm (R-) and for the cooperative strategy (R+). For dimension 10, next aspects should be highlighted:

- G-CMA-ES, CLPSO and JADE perform better than I and AC both in terms of number of functions with a higher performance and rank sum, although the null hypothesis can only be rejected for G-CMA-ES.
- Both I and AC improve MA-LSCh-CMA and GL-25. The improvement is significant between AC and GL-25.
Table 4: Comparison of I and AC with the participant of CEC2008 competition. CEC2008 participants are sorted according to their ranking in the competition from left to right.

<table>
<thead>
<tr>
<th>MTS</th>
<th>LSEDA-gl</th>
<th>jDEdynNP-F</th>
<th>MLCC</th>
<th>DMS-PSO</th>
<th>DEwSAcc</th>
<th>UEP</th>
<th>EPUS-PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worse</td>
<td>Better</td>
<td>Worse</td>
<td>Better</td>
<td>Worse</td>
<td>Better</td>
<td>Worse</td>
<td>Better</td>
</tr>
<tr>
<td>Rs</td>
<td>R-</td>
<td>Rs</td>
<td>R-</td>
<td>Rs</td>
<td>R-</td>
<td>Rs</td>
<td>R-</td>
</tr>
<tr>
<td>1</td>
<td>28</td>
<td>0</td>
<td>23</td>
<td>5</td>
<td>19</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>AC</td>
<td>7</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
<td>21</td>
<td>23</td>
<td>5</td>
<td>18</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td>4</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For dimension 30 the scenario is the following:

- MA-LSCh-CMA obtains a significant better performance than AC and I.
- G-CMA-ES and JADE show better results in a bigger number of functions and a higher rank sum than AC and I but the null hypothesis cannot be rejected.
- The performance of the two cooperative strategies is similar to the one obtain by CLPSO and GL-25.

Finally, when the dimension is 50 the main points to highlight are the next ones:

- The five state-of-the-art algorithms obtain a lower mean error in a bigger number of functions and a higher rank sum than the two strategies. However, the null hypothesis of equality cannot be rejected in none of the cases.
- The difference in performance among AC and the state-of-the-art algorithms is lower with dimension 50 than with dimension 30, excepting for GL-25.

The comparisons mentioned above indicate that I is some times more competitive than AC with respect to the state-of-the-art algorithms. This contrasts with the results seen in Section 5.1 where the independent method performed significantly worse than AC in an important number of functions. This is due to the different criteria followed by the Mann-Whitney’s and Wilcoxon’s paired non-parametric tests to establish the ranking of the algorithms. The reason why we have used different methodologies to compare the methods is the no availability of the error values obtained by some of the state-of-the-art algorithms in the 25 runs per function. Since in the literature their mean fitness value per function is only available, the best way to compare the performance is by means of a paired non-parametric test. The results seen above show that the proposed centralised cooperation and trajectory-based algorithms can obtain results that are competitive with state-of-the-art algorithms in continuous optimization.

As for the CEC2008 benchmark, the 8 algorithms that participated in that competition were [52]: MTS, LSEDA-gl, jDEdynNP-F, MLCC, DMS-PSO, DEwSAcc, UEP and EPUS-PSO. We will not describe these algorithms for the sake of the simplicity. In Table 4 the number of functions in which the mean error obtained by each version of the cooperative strategy is worse/better than the corresponding participant algorithm together with the rank sum returned by the Wilcoxon’s test are displayed. The methods in the columns are ordered, from left to right, according to their ranking in the competition. No information about the significance of the difference in performance is shown because the number of samples is too low to reject the null hypothesis in a robust way. Looking to this table, we can see that the first four CEC2008 algorithms clearly improve both I and AC. When this comparison is made with the four worst
participants, we can observe the best performance of the cooperative strategies specially AC. This means that our methods can obtain an intermediate position among algorithms that have been specifically designed to work with high dimensional functions despite of the fact that we are using the same configuration as in CEC2005 where the function sizes are much lower.

6. Analysing the behaviour induced by the cooperation

This section presents a new analysis to study the behaviour induced to the strategy by the different coordination modes. This analysis consists on studying the distribution of the local minima found by the cooperative strategy within the search space as well as their frequency of visiting. The proposal is based on Fitness Landscape Analysis and consists on analysing, on one hand, the distribution of the number of runs in which a local minima found within a region of the search space have been visited and on the other hand, the distribution of the mean frequency of visits per run of these local minima, both as a function of their fitness and the distance to the optimum solution. The first one allows us to check how the method explores the search space whereas the second one shows where the search is intensified. We have done this analysis for three functions of the CEC2005 benchmark: \(f_{10}, f_{22} \) and \(f_{25} \) with dimension 50. The two distributions for each methods over these functions are shown in Figures 4, 5 and 6, respectively. They collect information about the local minima found along the 25 runs and the steps done to build them are the next ones:

1. For every local minimum, the euclidean distance with respect to the global optimum \((d)\) is computed.
2. A grid is constructed where the \(Y\) axis represents the fitness of the local minima whereas \(X\) axis is used for distance \(d\). The length of both axis varies from minimum to maximum fitness or distance \(d\) and the grid is divided in 100 \(\times\) 100 cells of equal size.
3. Then for each method:
   (a) Compute the set of local minima \((lm_{ij})\) framed within each cell \((i, j)\) according to their fitness and \(d\) values.
   (b) Calculate a matrix \(runs\), where the component \(runs_{ij}\) corresponds to the number of distinct runs in which the local minima from \(lm_{ij}\) were visited \((runs_{ij})\).
(c) Calculate a matrix $\overline{\text{freq}}$, where $\overline{\text{freq}}_{ij}$ contains the mean frequency of visiting of the local optimum in $\text{lm}_{ij}$, that is, $\overline{\text{freq}}_{ij} = \frac{\sum_{x \in \text{lm}_{ij}} \varphi(x)}{\text{runs}_{ij}}$.

(d) The probability density function of runs and $\overline{\text{freq}}$ are then estimated by a two-dimensional kernel density estimation with an axis-aligned bivariate normal kernel.

For a better understanding of this process, a graphical example with a 3x3 grid, 9 local minima and 4 runs is given in Figure 3. The local minima are represented by a tuple where the first value indicates the run in which local minimum was found and the second value, the number of times it was visited in that run.

The probability density functions (PDFs) are represented by means of a grey scale that goes from light grey (low value) to black (high value). The figures show the PDFs for runs and $\overline{\text{freq}}$ on the left and right column, respectively. Apart from this, two contour lines matching points of equal probability are added to each plot in order to mark the “borders” of the PDF for the method’s (dotted line) and the independent strategy’s (straight line). The purpose of these two contour lines is to facilitate the visualization of the behaviour pattern induced by the cooperation respect to no-cooperation.

We will start this analysis with the function $f_{10}$ whose plots are displayed in Figure 4. Focusing on the left column, the plots show that the cooperative methods explore the search space in a different way to the independent strategy. This fact is especially clear for AC and RC since they are able to guide the solvers to regions closer to the global optimum. If we visualize the right column, the most interesting aspect is the narrowing suffered by the mean visiting frequency distributions of BSC, AC and RC. The interpretation is that these three actions concentrate their search on specific zones of the solution space, that is, they apply a bigger intensification to solvers, circumstance in accordance with the definition of these actions. The former distributions contrast with the ones of I and VRLC, more flattened, and which implies a more homogenous intensification in the visited regions due to their higher exploration balance.

Figure 5 presents the plots for $f_{22}$. A visual inspection to these graphs shows that the landscape of this function is completely different to the last one where there exists a certain correlation between the distance to the optimum and the fitness of the local minimum. In this case, when the distance is lower than 30 approximately, both variables are not correlated whereas for bigger distances, a negative correlation seem to appear (taking fitness as dependent variable). BSC and RC are the only methods that find better quality minima than I, as we can check in graphics for runs, indicating that locating the solvers around the best solution has a positive effect in this function. However, the characteristic of this optimization function are not suitable for AC that stagnate the search in local optima with a fitness similar to the best ones found by the independent strategy. As for VRLC, its higher exploration balance can be observed, although this bigger diversification takes place in regions with low quality solutions. Viewing the intensification patterns given by the mean frequency distributions, the differences between I and VRLC versus BSC, AC and RC are quite significant again. In this case, although the last three methods do not produce a narrowing in their probability density functions, the plots show how they intensify the search in the best fitness zones. On the contrary, VRLC and I present again a more uniform distribution along the explored space.

In the last function analysed, $f_{25}$, the cooperative methods found three basins of attraction, two from a similar distance from the optimum and one other farther. The plots for the independent method show us that the solvers are not able to “discover a path” between these basins of attraction but they get to reach them from a proper initial point. This behaviour takes also place for VRLC due to it restarts from randomly generated solutions. However, the other three
Figure 4: Plots for function $f_{10}$. Estimation, for I, BSC, AC, RC and VRLC, of the probability density functions (PDFs) of the number of runs in which a zone of the search space is visited (left column) and the mean frequency of visits per run (right column) as a function of their fitness and the distance to the global optimum. The two contour lines match points of equal probability and they mark the borders of the method’s (dotted line) and I strategy’s (straight line) PDFs.
Figure 5: Plots for function $f_{22}$. Estimation, for I, BSC, AC, RC and VRLC, of the probability density functions (PDFs) of the number of runs in which a zone of the search space is visited (left column) and the mean frequency of visits per run (right column) as a function of their fitness and the distance to the global optimum. The two contour lines match points of equal probability and they mark the borders of the method’s (dotted line) and I strategy’s (straight line) PDFs.
Figure 6: Plots for function $f_{25}$. Estimation, for I, BSC, AC, RC and VRLC, of the probability density functions (PDFs) of the number of runs in which a zone of the search space is visited (left column) and the mean frequency of visits per run (right column) as a function of their fitness and the distance to the global optimum. The two contour lines match points of equal probability and they mark the borders of the method’s (dotted line) and I strategies (straight line) PDF’s.
cooperation modes can find local optima between the basins of attraction because the solvers are brought near to the best solution. Regarding the performance, BSC is the method with more difficulties to reach good quality optima whereas I, VRLC and AC are actions that allow the strategy to visit the best basins in a higher number of runs. We will finish with the intensification patterns for this function. This follow a similar scheme to the two former functions where I and VRLC present a more uniform mean frequency of visiting, although in this case, BSC, AC and RC do not intensify the search in better fitness zones but in a big basin of attraction.

7. Conclusions

In this work, we have presented a centralised cooperative strategy for continuous optimisation problems whose solvers are all trajectory-based methods. The main motivations for the development of this strategy were the lack of this type of hybrid metaheuristics in this field, as far as our knowledge is concerned, and their success in other areas as combinatorial optimisation or dynamic optimisation problems. The strategy consisted on a set of solvers that implements a continuous tabu search with different parameter instantiations that are controlled by means of a rule-driven central coordinator.

Four control rules with different exploration/exploitation balance were considered on the experimentation done. These rules shared the same antecedent or condition but they have different consequents or actions. The action differs in how they reallocate, within the search space, a solver that is showing a bad performance. The proposed cooperative strategy working with each of these control rules as well as its independent version (the solvers do not exchange information) were evaluated on the benchmarks proposed for the CEC2005 competition on real parameter optimisation (dimensions 10, 30 and 50) and the CEC2008 competition on large scale optimisation (dimension 1000).

The results obtained showed that the four cooperative methods outperform the independent strategy in all the test suites considered. Comparing the coordination modes among them, we checked that AC was globally the best one, followed by RC, BSC and VRLC in that order. Another interesting issue of these results was the fact that some of control rules had complementary behaviours. The percentages of functions in which some pairs of cooperative methods improve each other were quite high and balanced.

The independent and the best performing version of the cooperative strategy (AC) were compared versus five state-of-the-art algorithms for the CEC2005 benchmark and versus the participants in the CEC2008 competition, obtaining competitive results in both cases. For CEC2005, only two of the five algorithms, G-CMA-ES and MACO-LSch-CMA, got to improve significantly the cooperative methods for dimensions 10 and 30, respectively, and furthermore, the hypothesis of equality respect to any of the five methods could not be rejected for dimension 50. In the second benchmark, I and AC were improved only by four of the eight participants (specifically designed for high dimensional problems) despite using the same configuration as for CEC2005.

Other contribution of this work was the proposal of an analysis inspired by Fitness Landscape Analysis ideas to study the behaviour induced by the cooperation in the strategy. This consisted on studying the distribution of the number of runs in which the local minima found within a region of the search space have been visited, on one hand, and the distribution of the mean frequency of visiting per run of these local minima, on the other hand. Both are given as a function of their fitness and the distance to the optimum solution. This analysis allows us to observe that the cooperation modes can lead the strategy to different behaviours which can be
more or less suitable for the characteristics of a determined function. Apart from this, we also checked that the diversification and intensification patterns of the strategy can be modified by the cooperation scheme.

From these promising results, several venues of research are opened. Firstly, in Section 5.1 we pointed out that the two actions with a better performance were those that use local and global information. Although this fact can only be considered as an indication, the benefit of taking into account the state of the solver or its local information is an issue that deserves further analysis.

Finally, we mentioned above that some rules have complementary behaviours, that is, the characteristics of the functions are more suitable for some actions than for others. In this sense, the incorporation into the coordinator of a mechanism that can infer the proper action for each function can lead the strategy to a noteworthy enhancement. To this end, the development of both off-line and on-line adaptation mechanisms to achieve this task will also focus our research in the future.

Acknowledgments

This work has been partially funded by the projects TIN2008-01948 and TIN2011-27696-C02-01 from the Spanish Ministry of Economy and Competitiveness, P07-TIC-02970 and P11-TIC-08001 from the Andalusian Government and FEDER funds from the European Union.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version.

References


