Comput Optim Appl (2013) 56:189–208
DOI 10.1007/s10589-013-9552-9

Multi Agent Collaborative Search based on Tchebycheff decomposition

Federico Zuiani · Massimiliano Vasile

Received: 16 January 2012 / Published online: 28 March 2013

Abstract This paper presents a novel formulation of Multi Agent Collaborative Search, for multi-objective optimization, based on Tchebycheff decomposition. A population of agents combines heuristics that aim at exploring the search space both globally (social moves) and in a neighborhood of each agent (individualistic moves). In this novel formulation the selection process is based on a combination of Tchebycheff scalarization and Pareto dominance. Furthermore, while in the previous implementation, social actions were applied to the whole population of agents and individualistic actions only to an elite subpopulation, in this novel formulation this mechanism is inverted. The novel agent-based algorithm is tested at first on a standard benchmark of difficult problems and then on two specific problems in space trajectory design. Its performance is compared against a number of state-of-the-art multi-objective optimization algorithms. The results demonstrate that this novel agent-based search has better performance with respect to its predecessor in a number of cases and converges better than the other state-of-the-art algorithms with a better spreading of the solutions.

Keywords Agent-based optimization · Multi-objective optimization · Memetic strategies

1 Introduction

Multi-Agent Collaborative Search (MACS) has been proposed as a framework for the implementation of hybrid, population-based, approaches for multi-objective opti-

F. Zuiani (✉)
School of Engineering, University of Glasgow, Glasgow, UK
e-mail: f.zuiani.1@research.gla.ac.uk

M. Vasile
Department of Mechanical & Aerospace Engineering, University of Strathclyde, Glasgow, UK
e-mail: Massimiliano.Vasile@strath.ac.uk

mization [20]. In this framework a number of heuristics are blended together in order to achieve a balanced global and local exploration. In particular, the search for Pareto optimal solutions is carried out by a population of agents implementing a combination of social and individualistic actions. An external archive is then used to reconstruct the Pareto optimal set.

The individualistic actions are devised to allow each agent to independently converge to the Pareto optimal set, thus creating its own partial representation of the Pareto front. Therefore, they can be regarded as memetic mechanisms associated to a single individual. The effectiveness of the use of local moves was recently demonstrated by Schuetze et al. [14]; Lara et al. [7] who proposed innovative local search mechanisms based on mathematical programming.

Other examples of memetic algorithms for multi-objective optimization use local sampling [5] or gradient-based methods [1–4, 6, 12, 16], generally building a scalar function to be minimized or hybridizing an evolutionary algorithm with a Normal Boundary Intersection (NBI) technique. The schedule with which the local search is run is critical and defines the efficiency of the algorithm.

MACS has been applied to a number of standard problems and real applications with good results, if compared to existing algorithms [10, 13, 18, 21]. The algorithm proposed in this paper is a novel version of Multi-Agent Collaborative Search, for multi-objective optimization problems, that implements some key elements of innovation. Most of the search mechanisms have been simplified but more importantly in this version Pareto dominance is not the only criterion used to rank and select the outcomes of each action. Instead, agents are using Tchebycheff decomposition to solve a number of single objective optimization problems in parallel. Furthermore, opposite to previous implementations of MACS, here all agents perform individualistic actions while social actions are performed only by selected sub-populations of agents.

Recent work by Zhang and Li [25] has demonstrated that Tchebycheff decomposition can be effectively used to solve difficult multi-objective optimization problems. Another recent example is Sindhya et al. [16] that uses Tchebycheff scalarization to introduce a local search mechanisms in NSGA-II. In this paper, it will be demonstrated how MACS based on Tchebycheff decomposition can achieve very good results on a number of cases, improving over previous implementations and state-of-the-art multi-objective optimization (MOO) algorithms.

The new algorithm is here applied to a set of known standard test cases and to two space mission design problems. The space mission design cases consider spacecraft equipped with a chemical engine and performing a multi-impulse transfer. They are part of a test benchmark for multi-impulsive problems that has been extensively studied in the single objective case but for which only a few comparative studies exist in the multi-objective case [11].

The paper is organized as follows: section two contains the general formulation of the problem with a brief introduction to Tchebycheff decomposition, the third section starts with a general introduction to the multi-agent collaborative search algorithm and heuristics before going into some of the implementation details. Section four contains a set of comparative tests that demonstrates the effectiveness of the new heuristics implemented in MACS. The section briefly introduces the performance metrics and ends with the results of the comparison.

2 Problem formulation

The focus of this paper is on finding the feasible set of solutions that solves the following problem:

$$\min_{\mathbf{x}\in D} \mathbf{f}(\mathbf{x}) \tag{1}$$

where D is a hyperrectangle defined as $D = \{x_j \mid x_j \in [b_j^l \; b_j^u] \subseteq \mathbb{R}, j = 1, \ldots, n\}$ and $\mathbf{f}$ is the vector function:

$$\mathbf{f} : D \to \mathbb{R}^m, \quad \mathbf{f}(\mathbf{x}) = \left[f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})\right]^T \tag{2}$$

The optimality of a particular solution is defined through the concept of dominance: with reference to problem (1), a vector $\mathbf{y} \in D$ is dominated by a vector $\mathbf{x} \in D$ if $f_l(\mathbf{x}) \leq f_l(\mathbf{y})$ for all $l = 1, \ldots, m$ and there exists k so that $f_k(\mathbf{x}) \neq f_k(\mathbf{y})$. The relation $\mathbf{x} \prec \mathbf{y}$ states that $\mathbf{x}$ dominates $\mathbf{y}$. A decision vector in D that is not dominated by any other vector in D is said to be Pareto optimal. All non-dominated decision vectors in D form the Pareto set D_P and the corresponding image in criteria space is the Pareto front

Starting from the concept of dominance, it is possible to associate, to each solution in a finite set of solutions, the scalar dominance index:

$$I_d(\mathbf{x}_i) = \left|\left\{i^* \;\middle|\; i, i^* \in N_p \wedge \mathbf{x}_{i^*} \prec \mathbf{x}_i\right\}\right| \tag{3}$$

where the symbol $|.|$ is used to denote the cardinality of a set and N_p is the set of the indices of all the solutions. All non-dominated and feasible solutions $\mathbf{x}_i \in D$ with $i \in N_p$ form the set:

$$X = \left\{\mathbf{x}_i \in D \mid I_d(\mathbf{x}_i) = 0\right\} \tag{4}$$

The set X is a subset of D_P, therefore, the solution of problem (1) translates into finding the elements of X. If D_P is made of a collection of compact sets of finite measure in $\mathbb{R}^n$, then once an element of X is identified it makes sense to explore its neighborhood to look for other elements of X. On the other hand, the set of non dominated solutions can be disconnected and its elements can form islands in D. Hence, multiple parallel exploration can increase the collection of elements of X.

2.1 Tchebycheff decomposition

In Tchebycheff' approach to the solution of problem (1), a number of scalar optimization problems are solved in the form:

$$\min_{\mathbf{x}\in D} g\left(\mathbf{f}(\mathbf{x}), \lambda, \mathbf{z}\right) = \min_{\mathbf{x}\in D} \max_{l=1,\ldots,m} \left\{\lambda_l \left| f_l(\mathbf{x}) - z_l \right|\right\} \tag{5}$$

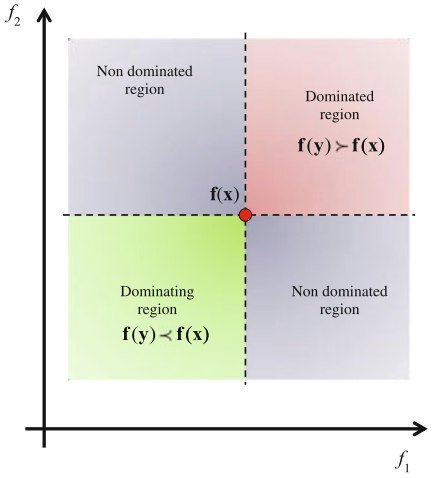
where $\mathbf{z} = [z_1, \ldots, z_m]^T$ is the reference objective vector whose components are $z_l = \min_{\mathbf{x}\in D} f_l(\mathbf{x})$, for $l = 1, \ldots, m$, and λ_l is the l-th component of the weight vector λ. By solving a number of problems (5), with different weight vectors, one can obtain different Pareto optimal solutions. Although the final goal is always to find the set X, using the solution of problem (5) or index (3) has substantially different consequences in the way samples are generated and selected. In the following, the solution to problem (5) will be used as selection criterion in combination with index (3).

3 MACS with Tchebycheff decomposition

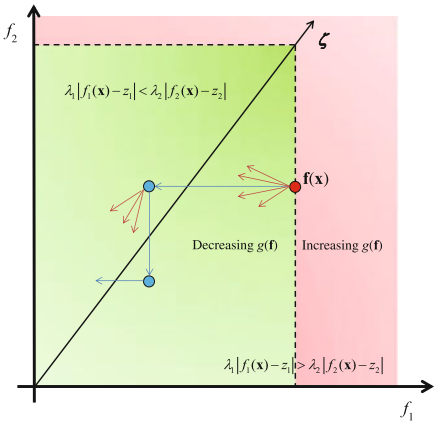
The key idea underneath multi-agent collaborative search is to combine local and global search in a coordinated way such that local convergence is improved while retaining global exploration [19]. This combination of local and global search is achieved by endowing a set of agents with a repertoire of actions producing either the sampling of the whole search space or the exploration of a neighborhood of each agent. Actions are classified into two categories: social, or collaborative, and individualistic. In this section, the key heuristics underneath MACS will be described in details. Compared to previous implementations of MACS [20], this paper proposes a number of key innovations. First of all, Tchebycheff decomposition is used in combination with dominance-based ranking to accept the outcome of an action. The idea is that each agent can either try to improve its dominance index or can try to improve one particular objective function by working on a subproblem characterized by a subset of weights λ. This combination extends the accepted individualistic moves and improves the spreading of the solutions in the criteria space. The second innovation comes from an inversion of the policy to schedule individualistic and social actions. In previous implementations the whole population was participating in the implementation of social actions at every generation, while an elite of agents was implementing individualistic actions. In this paper, this policy is inverted and now all the agents perform individualistic actions while selected subpopulations perform social actions either with other agents in the current population or with elements in the archive. This inversion is quite significant as it translates into a parallel local search performed by the whole population at each iteration, rather than having the local search performed by a selected number of individuals at a particular time of the evolution. More specific heuristics are described in the next sections.

The use of either dominance or Tchebycheff scalarization leads to the selection of different outcomes of the actions executed by the agents. With reference to Fig. 1(a) the dominance criterion can be used to select a displacement of agent $\mathbf{x}$ in the dominating region. In this case only strongly dominant solutions are accepted as admissible for a displacement of agent $\mathbf{x}$. Tchebycheff scalarization, instead, allows for movements in the region of decreasing $g(\mathbf{x})$ in Fig. 1(a).

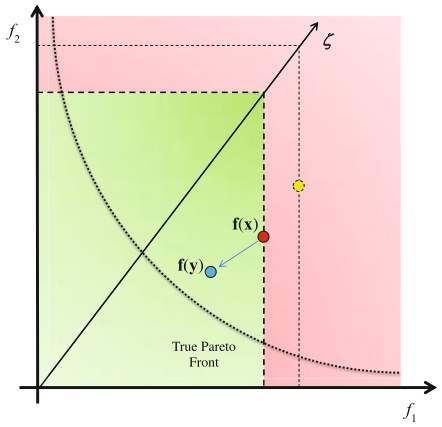
This region extends the dominating region of Fig. 1(a) and includes part of the non-dominating region. Therefore, Tchebycheff scalarization, as defined in (5) allows for the selection of weakly efficient solutions. If λ is kept constant the agent would progressively try to align along the direction ζ (see Fig. 1(b)). The rectilinear line ζ divides the criteria space in Fig. 1(b) in two half-planes, one, below ζ, where $\lambda_1|f_1(x) - z_1| > \lambda_2|f_2(\mathbf{x}) - z_2|$, the other, above ζ, where $\lambda_1|f_1(x) - z_1| < \lambda_2|f_2(\mathbf{x}) - z_2|$. The rectilinear line ζ is, therefore, the locus of points, in the criteria space, for which $\lambda_1|f_1(x) - z_1| = \lambda_2|f_2(\mathbf{x}) - z_2|$. Figure 1(b) shows that by solving problem (5) one would take displacements in any direction that improves f_1, starting from a solution that is under the ζ line. If one of these displacements crosses the ζ line, the solution of problem (5) would then generate displacements that improve f_2. This mechanisms allows for the generation of dominating steps (see Fig. 1(c)) as well as side steps (see Fig. 1(d)). Side steps are important to move along the Pareto front (see [7] for more details on the effect of side steps). In MACS side steps were gen-



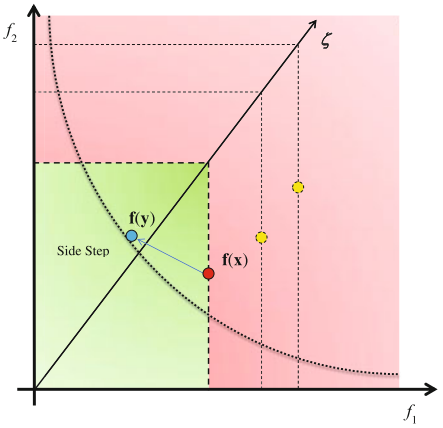
(a) Selection based on dominance index



(b) Selection based on Tchebycheff scalarization



(c) Selection based on Tchebycheff scalarization, strong dominance step



(d) Selection based on Tchebycheff scalarization, side step

Fig. 1 Selection criteria

erated by accepting displacements in the non-dominating regions of Fig. 1(a) when no dominant solutions were available. In MACS2 instead side steps are generated by selecting displacements according to Tchebycheff scalarization when strongly dominant solutions are not available. Note however, that although displacements are computed considering a combination of strong dominance and Tchebycheff scalarization, the archive is filled with all the solutions that have dominance index $I_d = 0$ and a large reciprocal distance (see Sect. 3.4).

3.1 General algorithm description

A population P_0 of n_{pop} virtual agents, one for each solution vector $\mathbf{x}_i$, with $i = 1, \ldots, n_{pop}$, is deployed in the problem domain D, and is evolved according to Algorithm 1.

The population P_h at iteration $h = 0$ is initialized using a Latin Hypercube distribution. Each agent then evaluates the associated objective vector $\mathbf{f}_i = \mathbf{f}(\mathbf{x}_i)$ and all non-dominated agents are cloned and inserted in the global archive A_g (lines 4 and 5 in Algorithm 1). The archive A_g contains the current best estimation of the target

Algorithm 1 MACS2

1: Set $n_{feval,max}$, n_{pop}, $n_{social} = round(\rho_{pop} n_{pop})$, F, tol_{conv}, $n_{A,out}$, u_{iter}
2: Set $n_\lambda = 100m$, $n_{A,max} = round(1.5 \max([n_\lambda, n_{A,out}]))$
3: Set $n_{feval} = 0$
4: Initialize population P_h, $h = 0$
5: Insert the non-dominated elements of P_0 in the global archive A_g
6: $\rho_i = 1$, $\forall i \in \{1, \ldots, n_{pop}\}$
7: Initialize λ_k for $k \in \{1, \ldots, n_\lambda\}$ such that $\|\lambda_k\| = 1$
8: Initialize utility function vector $U_k = 1$, $\forall k \in \{1, \ldots, n_\lambda\}$
9: Select the n_{social} active subproblems λ_l, and save their indexes l in the index set I_a
10: Initialize $\delta_l = \max_q \phi_{q,l} - \min_q \phi_{q,l}$, $z_l = \min_q \phi_{q,l}$, $q \in \{1, \ldots, |A_g|\}$, $l = 1, \ldots, m$,
11: **for all** $k \in \{1, \ldots, n_\lambda\}$ **do**
12: $\quad \underline{\phi}_k = \arg\min_{\phi_q} g(\phi_q, \lambda_k, \mathbf{z})$, $q = 1, \ldots, |A_g|$
13: **end for**
14: **for all** $\lambda_l, l \in I_a$ **do**
15: $\quad$ Select the $[\mathbf{x}_q \mathbf{f}_q] \in P_h$ which minimises $g(\mathbf{f}_q, \lambda_l, \mathbf{z}), l \in I_a$
16: $\quad$ and save its index in the list of the social agents I_λ
17: **end for**
18: **while** $n_{feval} < n_{feval,max}$ **do**
19: $\quad h = h + 1$
20: $\quad [P_h, n_{feval}, A_l, \rho] = explore(P_{h-1}, n_{feval}, n, \rho, \mathbf{b}^l, \mathbf{b}^u, \mathbf{f}, \lambda, I_\lambda, I_a)$
21: $\quad$ If necessary, update the vector of the best objectives $\mathbf{z}$, with A_l
22: $\quad$ Update archive A_g with non dominated elements of A_l
23: $\quad [\mathbf{y}, \varphi, n_{feval}, P_h, A_g] = com(P_h, A_g, \mathbf{b}^l, \mathbf{b}^u, n_{feval}, n, F, \mathbf{f}, \lambda, I_\lambda, I_a)$
24: $\quad$ **if** $|A_g| > n_{A,max}$ **then**
25: $\quad\quad A_g = resize(A_g, m, n_{A,max})$
26: $\quad$ **end if**
27: $\quad$ **if** (mod $(h, u_{iter}) = 0$) **then**
28: $\quad\quad [I_a, I_\lambda, \mathbf{U}, \underline{\phi}] = select(\mathbf{U}, \lambda, \underline{\phi}, P_k, A_g, \mathbf{z}, m, n_{social}, n_\lambda)$
29: $\quad$ **end if**
30: **end while**
31: $A_g = resize(A_g, m, n_{A,out})$

set X. The q-th element of the archive is the vector $\mathbf{a}_q = [\xi_q \ \phi_q]^T$ where ξ_q is a vector in the parameter space and ϕ_q is a vector in the criteria space.

Each agent is associated to a neighborhood D_{ρ_i} with size ρ_i. The size ρ_i is initially set to 1, i.e. representing the entire domain D (line 6 in Algorithm 1).

A set of n_λ, m-dimensional unit vectors λ_k is initialized such that the first m vectors are mutually orthogonal. The remaining $n_\lambda - m$ have random components instead. In two dimensions the vectors are initialized with a uniform sampling on a unit circle and in three dimensions with a uniform sampling on a unit sphere, while in n-dimensions with a Latin Hypercube sampling plus normalization, such that the length of each vector is 1 (see line 7 in Algorithm 1). For each vector λ_k, the value of an associated utility function U_k is set to 1 (see line 8 in Algorithm 1). The utility function is the one defined in [28] and its value is updated every u_{iter} iterations using Algorithm 5. In this work it was decided to maintain the exact definition and settings of the utility function as can be found in [28], the interested reader can therefore refer to [28] for further details.

Each λ_k represents a subproblem in Eq. (5), i.e. it is used to compute the scalar function g_k. A total of $n_{social} = round(\rho_{pop} n_{pop})$ λ vectors are inserted in the index set I_a. The first m indexes in I_a correspond to the m orthogonal λ vectors, the other $n_{social} - m$ are initially chosen randomly (line 9 of Algorithm 1).

Each λ_k for $k = 1, \ldots, n_\lambda$ is associated to the element in A_g that minimizes g_k such that:

$$\underline{\phi}_k = \arg\min_{\phi_q} g(\phi_q, \lambda_k, \mathbf{z}) \tag{6}$$

where $\mathbf{z}$ is the vector containing the minimum values of each of the objective functions. Then, for each λ_l, with $l \in I_a$ and associated vector $\underline{\phi}_l$, a *social* agent $\mathbf{x}_q$ is selected from the current population P_h such that it minimizes $g(\mathbf{f}_q, \lambda_l, \mathbf{z})$. The indexes of all the selected social agents are inserted in the index set I_λ (see lines 14 to 17 in Algorithm 1). The indexes in I_a and I_λ are updated every u_{iter} iterations.

At the h-th iteration, the population P_h is evolved through two sets of heuristics: first, every agent $\mathbf{x}_i$ performs a set of *individualistic actions* which aims at exploring a neighborhood D_{ρ_i} of $\mathbf{x}_i$ (line 20 of Algorithm 1), the function *explore* described in Algorithm 2 is used to implement individualistic actions. All the samples collected during the execution of individualistic actions are stored in the local archive A_l. The elements of A_l and the outcome of social actions are inserted in the global archive A_g if they are not dominated by any element of A_g (line 22 in Algorithm 1).

Then, a sub-population I_λ of n_{social} selected social agents performs a set of *social actions* (see line 23 of Algorithm 1). Social actions aim at sharing information among agents. More details about individualistic and social actions are provided in the following sections. The function *com* described in Algorithm 3 is used to implement social actions.

At the end of each iteration the global archive A_g is resized if its size has grown larger than $n_{A,max}$ (line 25 in Algorithm 1). The resizing is performed by function *resize* described in Algorithm 4.

Algorithm 2 explore—Individualistic Actions

$\Delta = (\mathbf{b}^u - \mathbf{b}^l)/2$
for all $i = 1 : n_{pop}$ **do**
  Set $A_{l,i} = \emptyset$, $p_i \in I_a$
  Take a random permutation I_E of $\{1, \ldots, n\}$
  for all $j \in I_E$ **do**
    Take a random number $r \in \mathcal{U}(-1, 1)$
    $\mathbf{y}^+ = \mathbf{x}_i$
    if $r > 0$ **then**
      $y_j^+ = \min\{y_j^+ + r\rho_i \Delta_j, b_j^u\}$
    else
      $y_j^+ = \max\{y_j^+ + r\rho_i \Delta_j, b_j^l\}$
    end if
    if $\mathbf{y}^+ \neq \mathbf{x}_i$ **then**
      Evaluate $\varphi^+ = \mathbf{f}(\mathbf{y}^+)$
      $n_{feval} = n_{feval} + 1$
      if $(\mathbf{y}^+ \not\succ \mathbf{x}_i)$ **then**
        $A_{l,i} = A_{l,i} \cup \{[\mathbf{y}^+ \varphi^+]\}$
      end if
      if $\mathbf{y}^+ \prec \mathbf{x}_i \vee (i \in I_\lambda \wedge g(\varphi^+, \lambda_{p_i}, \mathbf{z}) < g(\mathbf{f}_i, \lambda_{p_i}, \mathbf{z}))$ **then**
        $\mathbf{x}_i = \mathbf{y}^+$; break
      end if
    end if
    $\mathbf{y}^- = \mathbf{x}_i$
    Take a random number $rr \in \mathcal{U}(0, 1)$
    if $r > 0$ **then**
      $y_j^- = \max\{y_j^- - rr\rho_i \Delta_j, b_j^l\}$
    else
      $y_j^- = \min\{y_j^- + rr\rho_i \Delta_j, b_j^u\}$
    end if
    if $\mathbf{y}^- \neq \mathbf{x}_i$ **then**
      Evaluate $\varphi^- = \mathbf{f}(\mathbf{y}^-)$
      $n_{feval} = n_{feval} + 1$
      if $\mathbf{y}^- \not\succ \mathbf{x}_i$ **then**
        $A_{l,i} = A_{l,i} \cup \{[\mathbf{y}^- \varphi^-]\}$
      end if
      if $\mathbf{y}^- \prec \mathbf{x}_i \vee (i \in I_\lambda \wedge g(\varphi^-, \lambda_{p_i}, \mathbf{z}) < g(\mathbf{f}_i, \lambda_{p_i}, \mathbf{z}))$ **then**
        $\mathbf{x}_i = \mathbf{y}^-$; break
      end if
    end if
  end for
  if $\mathbf{y}^- \succ \mathbf{x}_i \wedge \mathbf{y}^+ \succ \mathbf{x}_i$ **then**
    $\rho_i = \eta_\rho \rho_i$
    if $\rho_i < tol_{conv}$ **then**
      $\rho_i = 1$
    end if
  end if
end for
$A_l = \bigcup_{i=1,\ldots,n_{pop}} A_{l,i}$

Algorithm 3 com—Social Actions

$p_{AvsP} = 1 - e^{-|A_g|/n_{social}}$
for all $i \in I_\lambda$ **do**
  $AvsP = r < p_{AvsP}$, $r \in \mathcal{U}(0,1)$, $p_i \in I_a$
  if $AvsP \wedge |A_g| \geq 3$ **then**
    Select the n_{social} closest elements of the archive A_g to the agent $\mathbf{x}_i$ and save their indexes in the set I_T
  else
    Select the n_{social} closest agents of the population P_k to the agent $\mathbf{x}_i$ and save their indexes in the set I_T
  end if
  $K \in \mathcal{U}(0,1)$
  Randomly select $s_1 \neq s_2 \neq s_3 \in I_T$
  $\mathbf{y} = \mathbf{x}_i + K(\mathbf{s}_3 - \mathbf{x}_i) + KF(\mathbf{s}_1 - \mathbf{s}_2)$
  for all $j \in \{1, \ldots, n\}$ **do**
    $r \in \mathcal{U}(0,1)$
    if $y_j < b_j^l$ **then**
      $y_j = b_j^l + r(y_j - b_j^l)$
    else if $y_j > b_j^u$ **then**
      $y_j = b_j^u - r(b_j^u - y_j)$
    end if
  end for
  if $\mathbf{y} \neq \mathbf{x}_i$ **then**
    Evaluate $\varphi = \mathbf{f}(\mathbf{y})$
    $n_{feval} = n_{feval} + 1$
  end if
  If necessary, update $\mathbf{z}$ with φ
  if $g(\varphi, \lambda_{p_i}, \mathbf{z}) < g(\mathbf{f}_i, \lambda_{p_i}, \mathbf{z})$ **then**
    $\mathbf{f}_i = \varphi$, $\mathbf{x}_i = \mathbf{y}$
  end if
  Update archive A_g with non-dominated elements of $\{[\mathbf{y}\ \varphi]\}$
end for

The value $n_{A,max}$ was selected to be the largest number between $1.5n_\lambda$ and $1.5n_{A,out}$, where $n_{A,out}$ is the desired number of Pareto optimal elements in A_g at the last iteration. This resizing of the archive is done in order to reduce the computational burden required by operations like the computation of the dominance index. It also provides an improved distribution of the solutions along the Pareto front as it discards solutions that are excessively cluttered.

At the end of each iteration the algorithm also checks if the maximum number of function evaluations $n_{feval,max}$, defined by the user, has been reached and if so, the algorithm terminates. At termination, the archive A_g is resized to $n_{A,out}$ if its cardinality is bigger than $n_{A,out}$.

Algorithm 4 resize—Archive Resizing

1: $n_A = |A_g|$, $S = \emptyset$
2: $\delta_j = \max_i \phi_{q,j} - \min_i \phi_{q,j}, \forall j = 1, \ldots, m$
3: **for all** $q \in \{1, \ldots, (n_A - 1)\}$ **do**
4: **for all** $i \in \{(q + 1), \ldots, n_A\}$ **do**
5: $d_{q,i} = \|(\phi_q - \phi_i)/\delta\|$
6: $d_{i,q} = d_{q,i}$
7: **end for**
8: **end for**
9: **for all** $l \in \{1, \ldots, m\}$ **do**
10: $S = S \cup \{\arg\min_q(\phi_{q,l})\}$
11: **end for**
12: $S_n = \{1, \ldots, n_A\} \setminus S$
13: **for all** $i \in \{m + 1, \ldots, n_{A,max}\}$ **do**
14: $l_S = \text{argmax}_l(\min_q(d_{q,l}))$, $q \in S$, $l \in S_n$
15: $S = S \cup \{l_S\}$
16: $S_n = S_n \setminus \{l_S\}$
17: **end for**
18: $A_g = \{\mathbf{a}_i | \forall i \in S\}$

3.2 Individualistic actions

Individualistic actions perform an independent exploration of the neighborhood D_{ρ_i} of each agent. As in the original version of MACS [18] the neighborhood is progressively resized so that the exploration is over the entire D when the size ρ_i is equal to 1 and becomes progressively more and more local as the neighborhood shrinks down. In this new implementation of MACS each agent performs only a simple sampling along the coordinates. The neighborhood D_{ρ_i} is a hypercube centered in $\mathbf{x}_i$ with size defined by ρ_i such that each edge of the hypercube has length $\rho_i(\mathbf{b}^u - \mathbf{b}^l)$. Algorithm 2 describes individualistic actions.

The search is performed along a single component of $\mathbf{x}_i$ at a time, in a random order: given an agent $\mathbf{x}_i$, a sample $\mathbf{y}^+$ is taken within D_{ρ_i} along the j-th coordinate with random step size $r \in \mathcal{U}(-1, 1)$, where $\mathcal{U}(-1, 1)$ is a uniform distribution over the closed interval $[-1\ 1]$, leaving the other components unchanged. If $\mathbf{y}^+$ dominates $\mathbf{x}_i$, $\mathbf{y}^+$ replaces $\mathbf{x}_i$, otherwise another sample $\mathbf{y}^-$ is taken in the opposite direction with step size rr, with $rr \in \mathcal{U}(0, 1)$. Again, if $\mathbf{y}^-$ dominates $\mathbf{x}_i$, $\mathbf{y}^-$ replaces $\mathbf{x}_i$. If $\mathbf{y}_i$ is not dominating and is not dominated by $\mathbf{x}_i$ and the index i of $\mathbf{x}_i$ belongs to I_λ, then $\mathbf{y}_i$ replaces $\mathbf{x}_i$ if $\mathbf{y}_i$ improves the value of the subproblem associated to $\mathbf{x}_i$. Whether a dominant sample or a sample that improves the value of the subproblem is generated the exploration terminates. This is a key innovation that exploits Tchebycheff decomposition and allows the agents to perform moves that improve one objective function at the time. The search terminates also when all the components of $\mathbf{x}_i$ have been examined, even if all the generated samples are dominated (see Algorithm 2 lines 3 to 40).

If all children are dominated by their parent, the size of the neighborhood ρ_i is reduced by a factor η_ρ. Finally, if ρ_i is smaller than a tolerance tol_{conv}, it is reset to 1

(see Algorithm 2 lines 41 to 46). In all the tests in this paper η_ρ was taken equal to 0.5 as this value provided good results, on average, across all test cases.

All the non-dominated children generated by each agent $\mathbf{x}_i$ during the exploration form the local archive $A_{l,i}$. The elements of $A_{l,i}$ are inserted in the global archive A_g if they are not dominated by any element in A_g.

3.3 Social actions

Social actions are performed by each agent whose index is in the set I_λ. Social actions are meant to improve the subproblem defined by the weight vectors λ_k in I_a and associated to the agents $\mathbf{x}_i$ in I_λ. This is done by exploiting the information carried by either the other agents in the population P_h or the elements in the archive A_g. Social actions implement the Differential Evolution (DE) heuristic:

$$\mathbf{y}_i = \mathbf{x}_i + K\big[(\mathbf{s}_1 - \mathbf{x}_i) + F(\mathbf{s}_2 - \mathbf{s}_3)\big] \tag{7}$$

where the vectors $\mathbf{s}_l$, with $l = 1, \ldots, 3$, are randomly taken from the local social network I_T of each social agent $\mathbf{x}_i$. The local social network is formed by either the n_{social} agents closest to $\mathbf{x}_i$ or the n_{social} elements of A_g closest to $\mathbf{x}_i$. The probability of choosing the archive vs. the population is directly proportional to p_{AvsP} (see line 3 of Algorithm 3). The parameter p_{AvsP} is defined as $1 - e^{-|A_g|/n_{social}}$. This means that in the limit case in which the archive is empty, the population is always selected. On the other hand, if the archive is much larger than the population, it is more likely to be selected. Note that, if the size of A_g is below 3 elements, then the population is automatically chosen instead (line 4 of Algorithm 3) as the minimum number of elements to form the step in (7) is 3. The offspring $\mathbf{y}_i$ replaces $\mathbf{x}_i$ if it improves the subproblem associated to $\mathbf{x}_i$ otherwise $\mathbf{y}_i$ is added to the archive A_g if it is not dominated by any of the elements of A_g. The value of F in this implementation is 0.9. Social actions, described in Algorithm 3, dramatically improve the convergence speed once a promising basin of attraction has been identified. On the other hand, in some cases social actions lead to a collapse of the subpopulation of social agents in one or more single points. This is in line with the convergence behavior of DE dynamics presented in [24]. This drawback is partially mitigated by the remaining agents which perform only *individualistic actions*. Algorithm 3 implements social actions.

3.4 Archive resizing

If the size of A_g exceeds a specified value (as detailed in Sect. 3.1), a resizing procedure is initiated. The resizing procedure progressively selects elements from the current archive and adds them to the resized archive until its specified maximum size $n_{A,max}$ is reached. First, the normalized Euclidean distances, in the objective space, between all the elements of the current archive are computed (lines 3–8 of Algorithm 4). Then the l-th element minimizing the l-th objective function, with $l = 1, \ldots, m$, is inserted in the resized archive (lines 9 to 12 of Algorithm 4). The remaining $n_{A,max} - m$ elements are iteratively selected by considering each time the element of the current archive (excluding those which are already in the resized one) which has the largest distance from its closet element in the resized archive (lines 13

Algorithm 5 select—Subproblem Selection

1: $\underline{\phi}_{old} = \underline{\phi}$
2: **for all** $k \in \{1, \ldots, n_\lambda\}$ **do**
3: $\quad \underline{\phi}_k = \arg\min_{\phi_q} g(\phi_q, \lambda_k, \mathbf{z}), q \in \{1, \ldots, |A_g|\}$
4: $\quad \gamma = (g(\underline{\phi}_{old,k}, \lambda_k, \mathbf{z}) - g(\underline{\phi}_k, \lambda_k, \mathbf{z}))$
5: $\quad$ **if** $\gamma > 0.001$ **then**
6: $\quad\quad U_k = 1$
7: $\quad$ **else**
8: $\quad\quad U_k = (0.95 + 50\gamma)U_k$
9: $\quad$ **end if**
10: **end for**
11: $t_{size} = round(n_\lambda/60)$
12: $I_a = \{1, \ldots, m\}$
13: **for all** $i \in \{m+1, \ldots, n_{social}\}$ **do**
14: $\quad$ Randomly select a subset I_{sel} of t_{size} elements of $\{1, \ldots, n_\lambda\}$
15: $\quad \bar{k} = \text{argmax}_k\, U_k, k \in I_{sel}$
16: $\quad I_a = I_a \cup \{\bar{k}\}$
17: **end for**
18: **for all** $\lambda_l, l \in I_a$ **do**
19: $\quad$ Select the $[\mathbf{x}_q \mathbf{f}_q] \in P_h$ which minimises $g(\mathbf{f}_q, \lambda_l, \mathbf{z}), l \in I_a$
20: $\quad$ and save its index in the list of the social agents I_λ
21: **end for**

to 17 of Algorithm 4). This procedure provides a good uniformity in the distribution of samples. Future work will investigate the comparative performance of different archiving strategies like the one proposed in [8] and [15].

3.5 Subproblem selection

Every u_{iter} iterations the active subproblems in I_a and the associated agents in I_λ performing *social* actions are updated. The agents performing social actions are updated through function *select* described in Algorithm 5.

The improvement γ between $\underline{\phi}_k$ (i.e. the best value of g_k at current iteration in the global archive) and $\underline{\phi}_{old,k}$ (the best value of g_k, u_{iter} iterations before) is calculated. Then, the utility function U_k associated to λ_k is updated according to the rule described in [28] and reported in Algorithm 5, lines 2 to 10.

Once a value U_k is associated to each λ_k, n_{social} new subproblems and associated λ vectors are selected. The first m λ vectors are always the orthogonal ones. The remaining $n_{social} - m$ are selected by taking $t_{size} = round(n_\lambda/60)$ random indexes and then choosing the one with the largest value of U_k. This is repeated till I_a is full (see lines 11 to 17 in Algorithm 5). Note that t_{size} cannot exceed the size of I_{tmp} in Algorithm 5 if the number of social agents n_{social} is small compared to n_λ

Finally, the agent $\mathbf{x}_i$, that minimizes the scalar objective function in Eq. (5) is associated to each λ_k with index in I_a, and its index is included in the new subset I_λ (lines 18 to 21 in Algorithm 5).

4 Experimental results

The new implementation of MACS is here called MACS2. This section presents the performance of MACS2 on a standard benchmark for multi-objective optimization algorithms and on some space-related test cases. Through an experimental analysis an optimal settings for MACS2 is derived. The results obtained with MACS2 will also be compared with those of MACS and other known multi-objective optimization algorithms [26]. The standard benchmark problems aim at optimizing the UF1-10 functions in the CEC09 test suite [27] and the test instances ZDT2, ZDT4, ZDT6 [29]. UF1 to UF7 are bi-objective test functions with 30 optimization parameters. UF8 to UF10 are tri-objective functions, again with 30 optimization parameters. The CEC09 competition rules specified 300000 function evaluations and 100 and 150 elements for the output Pareto fronts for the bi- and tri-objective functions respectively. ZDT2 ZDT4 and ZDT6 are bi-objective test cases with 30 parameters for the first one and 10 for the remaining two. They are tested running the algorithm for 25000 evaluations and taking an output front of 200 elements. The space-related test instances are given by two trajectory optimization problems as described in [11, 21]. The former is a 3-impulse transfer between a circular Low Earth Orbit (LEO) with radius $r_0 = 7000$ km to a Geostationary Orbit (GEO) with radius $r_f = 42000$ km. The latter test case, Cassini, describes a trajectory optimization instance from Earth to Jupiter with four intermediate gravity assists at Venus (twice), Earth and Jupiter respectively. For both test cases the objective functions to be minimized are total ΔV and time of flight. The 3-impulse test case has 5 optimization parameters and is run for 30000 function evaluations while Cassini has 6 parameters and is run for 600000 evaluations as it was demonstrated, in the single objective case, to have multiple nested local minima with a funnel structure [24]. The metrics which will be used in order to evaluate the performance of the algorithms are chosen so to have a direct comparison of the results in this paper with those in previous works. Therefore, for the CEC09 test set the IGD performance metric will be used [27]:

$$IGD\big(A, P^*\big) = \frac{1}{|P^*|} \sum_{\mathbf{v} \in P^*} \min_{\mathbf{a} \in A} \|\mathbf{v} - \mathbf{a}\| \tag{8}$$

where P^* is a set of equispaced points on the true Pareto front, in the objective space, while A is the set of points from the approximation of the Pareto front. As in [27], performance will be assessed as mean and standard deviation of the IGD over 30 independent runs. Note that a second batch of tests was performed taking 200 independent runs but the value of the IGD was providing similar indications. For the ZDT test set and for the space problems, the success rate on the convergence M_{conv} and spreading M_{spr} metrics are used instead. Note that, the IGD metric has been preferred for the UF test problems in order to keep consistency with the results presented in the CEC'09 competition. Convergence and spreading are defined as:

$$M_{conv} = \frac{1}{|A|} \sum_{\mathbf{a} \in A} \min_{\mathbf{v} \in P^*} \left\| \frac{\mathbf{v} - \mathbf{a}}{\delta} \right\| \tag{9}$$

$$M_{spr} = \frac{1}{|P^*|} \sum_{\mathbf{v} \in P^*} \min_{\mathbf{a} \in A} \left\| \frac{\mathbf{v} - \mathbf{a}}{\delta} \right\| \tag{10}$$

Table 1 Convergence tolerances

	3-impulse	Cassini	UF1	UF2	UF3	UF4	UF5	UF6
τ_{conv}	5e–2	7.5e–3	5e–3	5e–3	2e–2	3.5e–2	3e–2	3e–2
τ_{spr}	5e–2	5e–2	1e–2	1e–2	3e–2	3.5e–2	5e–2	3e–2

	UF7	UF8	UF9	UF10	ZDT2	ZDT4	ZDT6
τ_{conv}	5e–3	2e–2	3e–2	3e–2	1e–3	1e–2	1e–3
τ_{spr}	1e–2	6e–2	4e–2	6e–2	3e–3	1.5e–2	3e–3

Table 2 Reference settings for MACS2. Values within parenthesis are for 3-impulse and ZDT test cases

n_{pop}	ρ_{pop}	F	Tol_{conv}
60 (30)	0.33	0.5	0.0001

with $\delta = \max_i \mathbf{a}_{f,i} - \min_i \mathbf{a}_{f,i}$. It is clear that M_{spr} is the IGD but with the solution difference, in objective space, normalized with respect to the exact (or best-so-far) solution. In the case of the ZDT test set, the two objective functions range from 0 to 1, therefore no normalization is required and M_{spr} is in fact the IGD. The success rates for M_{conv} and M_{spr} is defined as $p_{conv} = P(M_{conv} < \tau_{conv})$ and $p_{spr} = P(M_{spr} < \tau_{spr})$ respectively, or the probability that the indexes M_{conv} and M_{spr} achieve a value less than the threshold τ_{conv} and τ_{spr} respectively. The success rates p_{conv} and p_{spr} are computed over 200 independent runs, hence they account for the number of times M_{conv} and M_{spr} are below their respective thresholds. According to the theory developed in [11, 23], 200 runs provide a 5 % error interval with a 95 % confidence level. Values for thresholds for each test case are reported in Table 1.

MACS2 was initially set with a some arbitrary values reported in Table 2. The size of the population was set to 60 for all the test cases except for the 3-impulse and ZDT functions. For these test cases the number of agents was set to 30. In the following, these values will identify the *reference* settings.

Starting from this reference settings a number of tuning experiments were run to investigate the reciprocal influence of different parameters and different heuristics within the algorithm. Different combinations of n_{pop}, ρ_{pop}, F and Tol_{conv} were considered. Furthermore, the social moves were activated or de-activated to assess their impact. The success rates were then used to tune the algorithm in order to improve the spreading, and therefore the IGD. After an extensive testing of the algorithms, it was realized that the use of the success rates offers a clearer metric, than the mean and variance of the IGD, to understand the impact of some user-defined parameters. In the following, only the most significant results with the most significant metric are presented.

Table 3 summarizes the success rates on the Cassini test case for different values of n_{pop} and ρ_{pop} but with all the heuristics active.

One can see that the best convergence is obtained for $n_{pop} = 150$ and in particular when combined with $\rho_{pop} = 0.5$. On the other hand, best spreading is obtained with medium sized populations with $n_{pop} = 60$. A good compromise seems to be $n_{pop} =$

Table 3 Tuning of n_{pop} and ρ_{pop} on the Cassini test case

p_{conv}			
$\rho_{pop} \backslash n_{pop}$	20	60	150
0.2	0.22	0.34	0.76
0.5	0.16	0.41	0.78
0.8	0.35	0.40	0.77
p_{spr}			
$\rho_{pop} \backslash n_{pop}$	20	60	150
0.2	0.32	0.45	0.31
0.5	0.45	0.48	0.26
0.8	0.37	0.40	0.26

Table 4 Tuning of MACS2 on the 3-impulse and Cassini test cases

	3-impulse		Cassini	
	p_{conv}	p_{spr}	p_{conv}	p_{spr}
Reference	0.99	0.99	0.38	0.36
no social	0.47	1	0	0.18
$n_{pop} = 150$, $\rho_{pop} = 0.2$	1	1	0.76	0.31
$F = 0.9$	0.97	0.99	0.50	0.36
$Tol_{conv} = 10^{-6}$	0.99	0.99	0.38	0.45
$Tol_{conv} = 10^{-2}$	0.97	0.99	0.33	0.39

150 and $\rho_{pop} = 0.2$. Results on the other test cases (as shown in Table 4, Table 5 and Table 6, with $n_{pop} = 150$ and $\rho_{pop} = 0.2$) show in general that large populations and small ρ_{pop} are preferable. This also means that social actions on a large quota of the populations are undesirable and it is better to perform social moves among a restricted circle of agents. Table 4 reports the results of the tuning of MACS2 on the 3-imp and Cassini test cases. Table 5 and Table 6 report the results of the tuning of MACS2 on the UF and ZDT test sets respectively.

Table 4 shows a marked improvement of p_{conv} on the Cassini when the population size is 150. Likewise, Table 5 shows that in general, with a population of 150 agents, there is an improvement in performance, and on p_{spr} in particular, on the UF1, 2, 6, 8 and 9 test cases. Notable exceptions are the ZDT in Table 6, for which the best performance is obtained for a small population with $n_{pop} = 20$.

The impact of F is uncertain in many cases, however, Table 7 shows for example that on the UF8 test case a better performance is obtained for a high value of F. Table 5 and Table 6 show that the default value for Tol_{conv} already gives good performance and it does not seem advantageous to reduce it or make it larger.

The impact of social actions can be seen in Table 4, Table 5 and Table 6. Table 4 shows that on the 3-impulse and Cassini test cases the impact is clearly evident, since there is a marked worsening of both p_{conv} and p_{spr}. On the UF benchmark, see Table 5, removing social actions induces a sizeable worsening of the performance metrics. This is true in particular for functions UF1, UF3, UF5, UF6, UF7, UF8 and

Table 5 Tuning of MACS2 on the UF test cases

		Reference	no social	$n_{pop} = 150$ $\rho_{pop} = 0.2$	$n_{pop} = 20$ $\rho_{pop} = 0.8$	$Tol_{conv} = 10^{-6}$
UF1	p_{conv}	1	1	1	1	1
	p_{spr}	1	1	1	0.11	1
UF2	p_{conv}	1	1	1	1	1
	p_{spr}	1	1	1	0.46	1
UF3	p_{conv}	0.95	0.32	0.99	0.86	0.95
	p_{spr}	0.99	0.11	1	0.97	1
UF4	p_{conv}	1	1	1	0.06	1
	p_{spr}	1	1	1	0.54	1
UF5	p_{conv}	0.59	0.10	0.62	0.91	0.58
	p_{spr}	0.85	0.21	1	0.39	0.85
UF6	p_{conv}	0.58	0.50	0.32	0.54	0.61
	p_{spr}	0.40	0.42	0.45	0	0.37
UF7	p_{conv}	1	0.91	1	0.94	1
	p_{spr}	0.98	0	0.98	0.74	0.97
UF8	p_{conv}	0.86	0	0.88	0.89	0.88
	p_{spr}	0.48	0.01	1	0.04	0.54
UF9	p_{conv}	0.68	0.12	0.84	0.31	0.74
	p_{spr}	0.60	0	1	0	0.64
UF10	p_{conv}	0	0.01	0	0.28	0.01
	p_{spr}	0	0	0	0	0

Table 6 Tuning of MACS2 on ZDT test cases

		ZDT2 $\tau_{conv} = 1e{-}3$ $\tau_{spr} = 3e{-}3$	ZDT4 $\tau_{conv} = 1e{-}2$ $\tau_{spr} = 1.5e{-}2$	ZDT6 $\tau_{conv} = 1e{-}3$ $\tau_{spr} = 3e{-}3$
Reference	p_{conv}	1	0	0.93
	p_{spr}	1	0	1
no social	p_{conv}	1	0	0.91
	p_{spr}	1	0	0.98
$n_{pop} = 150$	p_{conv}	0.20	0	0.60
$\rho_{pop} = 0.2$	p_{spr}	0.17	0	1
$n_{pop} = 20$	p_{conv}	1	0.02	0.96
$\rho_{pop} = 0.8$	p_{spr}	1	0.02	1
$F = 0.9$	p_{conv}	1	0	0.96
	p_{spr}	1	0	1
$Tol_{conv} = 1e{-}6$	p_{conv}	1	0	0.96
	p_{spr}	1	0	1
MACS2 (Tuned)	p_{conv}	1	0	0.96
	p_{spr}	1	0	1
MACS	p_{conv}	0.82	0.81	0.63
	p_{spr}	0	0.93	0.0

Table 7 Tuning of F on the UF8 test cases

UF8			
F	0.1	0.5	0.9
IGD	6.75e–2 (3.20e–5)	6.06e–2 (2.56e–5)	5.57e–2 (1.87e–5)

Table 8 Settings for MACS2 after tuning

n_{pop}	ρ_{pop}	F	Tol_{conv}
150(20)	0.2(0.8)	0.9	10^{-4}

Table 9 Performance comparison on UF test cases: Average IGD (variance within parenthesis)

	MACS2	MACS	MOEAD	MTS	DMOEADD
UF1	4.37e–3 (1.67e–8)	1.15e–1 (1.66e–3)	**4.35e–3**	6.46e–3	1.04e–2
UF2	**4.48e–3 (1.16e–8)**	5.43e–2 (4.19e–4)	6.79e–3	6.15e–3	6.79e–3
UF3	**2.29e–2 (5.21e–6)**	6.56e–2 (1.42e–3)	7.42e–3	5.31e–2	3.34e–2
UF4	2.64e–2 (3.48e–7)	3.36e–2 (1.66e–5)	6.39e–2	**2.36e–2**	4.27e–2
UF5	2.95e–2 (1.56e–5)	6.44e–2 (1.17e–3)	1.81e–1	**1.49e–2**	3.15e–1
UF6	**3.31e–2 (7.42e–4)**	2.40e–1 (1.43e–2)	1.76e–1	5.91e–2	6.67e–2
UF7	6.12e–3 (3.14e–6)	1.69e–1 (1.22e–2)	**4.44e–3**	4.08e–2	1.03e–2
UF8	**4.98e–2 (2.05e–6)**	2.35e–1 (1.77e–3)	5.84e–2	1.13e–1	6.84e–2
UF9	**3.23e–2 (2.68e–6)**	2.68e–1 (1.71e–2)	7.90e–2	1.14e–1	4.90e–2
UF10	**1.41e–1 (5.59e–5)**	1.25 (4.28e–1)	4.74e–1	1.53e–1	3.22e–1

UF9. Notable exceptions are UF2, UF4 and UF10. As a results of the tuning test campaign, the settings reported in Table 8 are recommended. Note that the recommended population size for all the cases except the ZDT functions, is 150 agents, while for the ZDT functions remains 20 agents.

With these settings, the performance of MACS2 was compared, on the UF test suite in Table 9, with that of MACS, Multi objective Evolutionary Algorithm based on Decomposition (MOEAD [25]), Multiple Trajectory Search (MTS [17]) and Dynamical Multi Objective Evolutionary Algorithm (DMOEADD [9]). The last three are the best performing algorithms in the CEC09 competition [26].

As shown in Table 9, the tuned version of MACS2 outperforms the other algorithms on UF2, 3, 6, 8, 9 and 10, on UF1 is very close to MOEAD, while it ranks second on UF5 and 10 and finally third on UF7.

In Table 6 one can find the comparison against the old version MACS on the ZDT test set. MACS2 results generally better except on the ZDT4 case. Note that M_{spr} of MACS for both ZDT2 and ZDT6 is always between 0.6e-2 and 0.9e-2, therefore always above the chosen threshold τ_{spr}.

The poor performance of MACS2 on ZDT4, might be due to the relative ineffectiveness of the pattern search along the coordinates on this particular test case. In the attempt to improve performance on ZDT4, a second test set was run with a slightly modified version of MACS2: the number of components which are explored by each

Table 10 Comparison of MACS, MACS2 and MOEAD on 3-impulse and Cassini test cases

	3-impulse		Cassini	
	p_{conv}	p_{spr}	p_{conv}	p_{spr}
MACS	0.99	0.99	0.87	0.49
MACS2 (Tuned)	0.99	1	0.77	0.34
MOEAD	1	0.49	0.51	0.01
MTS	0.57	1	0.05	0.32
NSGA-II	0.03	1	0.90	0.26

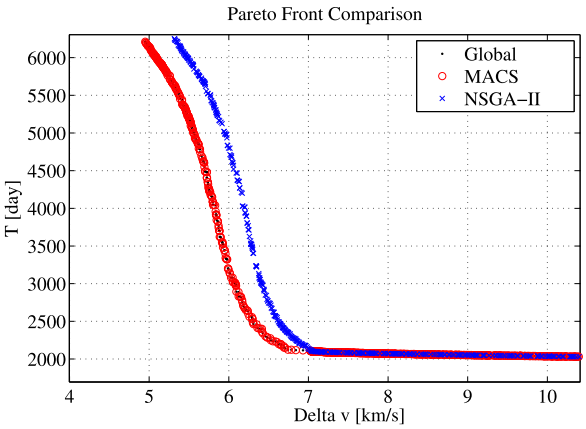


Fig. 2 Comparison of Pareto fronts for the Cassini case

agent at the h-th iteration was reduced to 1 only, compared to the n in Algorithm 2, at the same time, all individuals were performing social actions, i.e. $n_{social} = n_{pop}$. With this modifications, a success rate of 0.66 both on convergence and spreading is achieved although the p_{conv} and p_{spr} on ZDT2 drops to 0 and the p_{conv} on ZDT6 drops to 23 %.

Table 10 shows a comparison of the performance of MACS2 on 3-impulse and Cassini, against MACS, MOEAD, MTS and NSGA-II. Both MACS and MACS2 are able to reliably solve the 3-impulse case, while MOEAD manages to attain good convergence but with only mediocre spreading. On the contrary, both MTS and NSGA-II achieve good spreading but worse convergence, indicating that their fronts are quite well distributed but probably too distant from the true Pareto front. Cassini is a rather difficult problem and this is reflected by the generally lower metrics achieved by most algorithms. Only MACS, MACS2 and NSGA-II reach a high convergence ratio, but for the last two, their spreading is still rather low. After inspection of each of the 200 Pareto fronts one can see that such a low spreading implies that the algorithm did not converge to the global Pareto front. Figure 2 illustrates the difference between MACS and NSGA-II. The behavior of MACS2 is similar to the one of NSGA-II. MACS achieves the best known value for objective function Δv. Both NSGA-II and MACS2 instead fall in the basin of attraction of the second best value for objective function Δv [22].

The performance of MOEAD and MTS on Cassini is rather poor, with the former attaining only 50 % convergence but with almost zero p_{spr}; conversely, only one third of the latter's runs are below the spreading threshold and almost none meets the convergence criterion.

5 Conclusions

This paper has presented a version of Multi-Agent Collaborative Search based on Tchebycheff decomposition. Compared to the previous version of MACS a number of heuristics has been revised and in particular there was an inversion of the percentage of agents performing social and individualistic moves. The new version, denominated MACS2, demonstrated remarkable performance on known difficult benchmarks outperforming known algorithms. On the Cassini real case application, and on benchmark function ZDT4, MACS2 falls back behind its predecessor. In both cases there are multiple local Pareto fronts corresponding to strong attractors. From a first analysis it seems that the simple pattern search implemented in MACS2 is not sufficient and is limited by its search along the coordinates only. In MACS the search included random directions and directions derived from DE and PSO heuristics. It seems reasonable to assume that a more flexible set of individualistic moves might improve MACS2. This is the subject of current developments. Also, from the tests performed so far the actual contribution of the utility function is uncertain and more investigations are underway.

The use of a selection operator based on Tchebycheff decomposition, instead, appears to be beneficial in a number of cases. In MACS2, in particular, agents operating at the extreme of the range of each objective are always preserved and forced to improve a subproblem. A better solution of the subproblems is expected to further improve convergence. One possibility currently under investigation is to make some agents use a directed search exploiting the directions defined by the λ vectors.

References

1. Erfani, T., Utyuzhnikov, S.: Directed search domain: a method for even generation of the Pareto frontier in multiobjective optimization. Eng. Optim. **43**(5), 467–484 (2011)
2. Fliege, J., Drummond, M., Svaiter, B.: Newton's method for multicriteria optimization. SIAM J. Optim. **20**(2), 602–626 (2009)
3. Graña Drummond, L., Svaiter, B.: A steepest descent method for vector optimization. J. Comput. Appl. Math. **175**(2), 395–414 (2005)
4. Ishibuchi, H., Yoshida, T.: Hybrid evolutionary multi-objective optimization algorithms. In: Soft Computing Systems: Design, Management and Applications, pp. 163–172. IOS Press, Amsterdam (2002)
5. Knowles, J., Corne, D.: Local search, multiobjective optimization and the Pareto archived evolution strategy. In: Proceedings of Third Australia-Japan Joint Workshop on Intelligent and Evolutionary Systems, Ashikaga Institute of Technology, pp. 209–216 (1999). http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.33.6848
6. Kumar, A., Sharma, D., Deb, K.: A hybrid multi-objective optimization procedure using PCX based NSGA-II and sequential quadratic programming. In: IEEE Congress on Evolutionary Computation (CEC 2007), pp. 3011–3018. IEEE Press, New York (2007)
7. Lara, A., Sanchez, G., Coello Coello, C., Schutze, O.: HCS: a new local search strategy for memetic multiobjective evolutionary algorithms. IEEE Trans. Evol. Comput. **14**(1), 112–132 (2010)

8. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multiobjective optimization. Evol. Comput. **10**(3), 263–282 (2002)
9. Liu, M., Zou, X., Chen, Y., Wu, Z.: Performance assessment of DMOEA-DD with CEC 2009 MOEA competition test instances. In: IEEE Congress on Evolutionary Computation (CEC'09), pp. 2913–2918. IEEE Press, New York (2009)
10. Maddock, C., Vasile, M.: Design of optimal spacecraft-asteroid formations through a hybrid global optimization approach. Int. J. Intell. Comput. Cybern. **1**(2), 239–268 (2008)
11. Minisci, E., Avanzini, G.: Orbit transfer manoeuvres as a test benchmark for comparison metrics of evolutionary algorithms. In: IEEE Congress on Evolutionary Computation, 2009. CEC'09, pp. 350–357. IEEE Press, New York (2009)
12. Rigoni, E., Poles, S.: NBI and MOGA-II, two complementary algorithms for multi-objective optimizations. In: Practical Approaches to Multi-objective Optimization, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany (2005). http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.89.6798
13. Sanchez, J., Colombo, C., Vasile, M., Radice, G.: Multi-criteria comparison among several mitigation strategies for dangerous near earth objects. J. Guid. Control Dyn. **32**(1), 121–142 (2009)
14. Schuetze, O., Sanchez, G., Coello Coello, C.: A new memetic strategy for the numerical treatment of multi-objective optimization problems. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, pp. 705–712. ACM, New York (2008)
15. Schütze, O., Laumanns, M., Tantar, E., Coello, C., Talbi, E.: Computing gap free Pareto front approximations with stochastic search algorithms. Evol. Comput. **18**(1), 65–96 (2010)
16. Sindhya, K., Sinha, A., Deb, K., Miettinen, K.: Local search based evolutionary multi-objective optimization algorithm for constrained and unconstrained problems. In: IEEE Congress on Evolutionary Computation (CEC'09), pp. 2919–2926. IEEE Press, New York (2009)
17. Tseng, L., Chen, C.: Multiple trajectory search for unconstrained/constrained multi-objective optimization. In: IEEE Congress on Evolutionary Computation (CEC'09), pp. 1951–1958. IEEE Press, New York (2009)
18. Vasile, M.: Robust mission design through evidence theory and multiagent collaborative search. Ann. N.Y. Acad. Sci. **1065**(1), 152–173 (2005)
19. Vasile, M., Locatelli, M.: A hybrid multiagent approach for global trajectory optimization. J. Glob. Optim. **44**(4), 461–479 (2009)
20. Vasile, M., Zuiani, F.: A hybrid multiobjective optimization algorithm applied to space trajectory optimization. In: IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE Press, New York (2010)
21. Vasile, M., Zuiani, F.: Multi-agent collaborative search: an agent-based memetic multi-objective optimization algorithm applied to space trajectory design. Proc. Inst. Mech. Eng., G J. Aerosp. Eng. **225**(11), 1211–1227 (2011)
22. Vasile, M., Minisci, E., Locatelli, M.: A dynamical system perspective on evolutionary heuristics applied to space trajectory optimization problems. In: IEEE Congress on Evolutionary Computation (CEC'09), pp. 2340–2347. IEEE Press, New York (2009)
23. Vasile, M., Minisci, E., Locatelli, M.: Analysis of some global optimization algorithms for space trajectory design. J. Spacecr. Rockets **47**(2), 334–344 (2010)
24. Vasile, M., Minisci, E., Locatelli, M.: An inflationary differential evolution algorithm for space trajectory optimization. IEEE Trans. Evol. Comput. **15**(2), 267–281 (2011)
25. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **11**(6), 712–731 (2007)
26. Zhang, Q., Suganthan, P.: Final report on CEC'09 MOEA competition. In: IEEE Congress on Evolutionary Computation (CEC'09) (2009)
27. Zhang, Q., Zhou, A., Zhao, S., Suganthan, P., Liu, W., Tiwari, S.: Multiobjective optimization test instances for the CEC 2009 special session and competition. Technical Report, University of Essex, Colchester, UK and Nanyang Technological University, Singapore, Special Session on Performance Assessment of Multi-Objective Optimization Algorithms (2008)
28. Zhang, Q., Liu, W., Li, H.: The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. In: IEEE Congress on Evolutionary Computation (CEC'09), pp. 203–208. IEEE Press, New York (2009)
29. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans. Evol. Comput. **7**(2), 117–132 (2003)