



AUTHOR(S):

TITLE:

YEAR:

Publisher citation:

OpenAIR citation:

Publisher copyright statement:

This is the _____ version of an article originally published by _____
in _____
(ISSN _____; eISSN _____).

OpenAIR takedown statement:

Section 6 of the "Repository policy for OpenAIR @ RGU" (available from <http://www.rgu.ac.uk/staff-and-current-students/library/library-policies/repository-policies>) provides guidance on the criteria under which RGU will consider withdrawing material from OpenAIR. If you believe that this item is subject to any of these criteria, or for any other reason should not be held on OpenAIR, then please contact openair-help@rgu.ac.uk with the details of the item and the nature of your complaint.

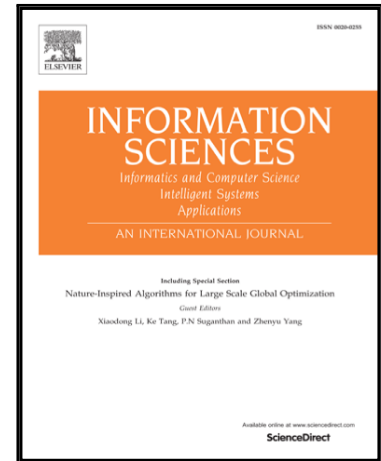
This publication is distributed under a CC _____ license.

Accepted Manuscript

Region-based Memetic Algorithm with Archive for multimodal optimisation

Benjamin Lacroix, Daniel Molina, Francisco Herrera

PII: S0020-0255(16)30388-7
DOI: [10.1016/j.ins.2016.05.049](https://doi.org/10.1016/j.ins.2016.05.049)
Reference: INS 12270



To appear in: *Information Sciences*

Received date: 5 March 2015
Revised date: 13 April 2016
Accepted date: 29 May 2016

Please cite this article as: Benjamin Lacroix, Daniel Molina, Francisco Herrera, Region-based Memetic Algorithm with Archive for multimodal optimisation, *Information Sciences* (2016), doi: [10.1016/j.ins.2016.05.049](https://doi.org/10.1016/j.ins.2016.05.049)

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

Region-based Memetic Algorithm with Archive for multimodal optimisation

Benjamin Lacroix^a, Daniel Molina^b, Francisco Herrera^{c,d}

^a IDEAS Research Institute, Robert Gordon University, United Kingdom

^b Department of Computer Science, University of Cádiz, 11003, Cádiz, Spain

^c Department of Computer Science and Artificial Intelligence, University of Granada, 18071, Granada, Spain

^d Faculty of Computing and Information Technology, King Abdulaziz University, 21589, Jeddah, Saudi Arabia

Abstract

In this paper we propose a specially designed memetic algorithm for multimodal optimisation problems. The proposal uses a niching strategy, called region-based niching strategy, that divides the search space in predefined and indexable hypercubes with decreasing size, called regions. This niching technique allows our proposal to keep high diversity in the population, and to keep the most promising regions in an external archive. The most promising solutions are improved with a local search method and also stored in the archive. The archive is used as an index to efficiently prevent further exploration of these areas with the evolutionary algorithm. The resulting algorithm, called Region-based Memetic Algorithm with Archive, is tested on the benchmark proposed in the special session and competition on niching methods for multimodal function optimisation of the Congress on Evolutionary Computation in 2013. The results obtained show that the region-based niching strategy is more efficient than the classical niching strategy called clearing and that the use of the archive as restrictive index significantly improves the exploration efficiency of the algorithm. The proposal achieves better exploration and accuracy than other existing techniques.

Keywords: Multimodal optimisation, memetic algorithm, niching strategy

1. Introduction

Many real world problems offer various solutions considered as global optima. The identification of multiple solution has thus gained popularity in the research community. It is referred to as multimodal optimisation as the objective is to retrieve more than one optima. While classical evolutionary algorithms

Email addresses: b.m.e.lacroix@rgu.ac.uk (Benjamin Lacroix), daniel.molina@uca.es (Daniel Molina), herrera@decsai.ugr.es (Francisco Herrera)

6 (EA) were designed to identify a single optimum, some modifications have to
7 be applied to identify multiple optima, preventing their premature convergence
8 and maintaining the diversity in their population to ensure the exploration of
9 distinct areas of the fitness landscape. Such techniques, known as niching strate-
10 gies [6], are meant to stay in the population subgroups of individuals, or *niches*,
11 in different parts of the search domain.

12 Most existing techniques' efficiency relies on two problem dependent param-
13 eters, the niche radius and the population size [7, 16, 42]. The first one should
14 be defined according to the distance between optima in the fitness landscape
15 and the second one according to the number of optima to locate. Both data are
16 however usually unknown in real world problems. Nowadays, research interest
17 focuses on designing EA which are less dependent on those parameters.

18 The main challenge when designing an EA for multimodal optimisation is to
19 create an algorithm capable of approximating with the highest level of accuracy
20 the different global optima.

21 Memetic algorithms (MA) [35] are the hybridisation between EA and local
22 search methods (LS) combining in one model the exploration power of the for-
23 mer and the exploitation capacity of the latter. This hybridisation can achieve
24 a good trade-off between the exploration of the domain search and the exploita-
25 tion of found solutions, so it is important to obtain good results in EAs [59],
26 and it also offers interesting properties when applying them to multimodal opti-
27 misation problems from the multimodal optimisation point of view. Indeed, as
28 we said before, niching techniques used with classical EA forms sub-populations
29 destined to explore and optimise different areas of the search space with the
30 same mechanism. MA separate these efforts, leaving the exploration task to the
31 EA and the refinement of the most promising regions identified by the EA to
32 the LS method.

33 In a previous work [21], we designed a MA for global continuous optimisa-
34 tion problems called region based memetic algorithm with local search chaining
35 (RMA-LSCh). It proposed a novel niching strategy, the originality of which
36 lies in the definition of a niche. While traditionally the niche surrounding a
37 solution is defined by the radius around it, the proposed niching technique par-
38 titions the search into equal hypercubes called regions. The dependency to the
39 niche size (defined by the number of divisions of the search space) is reduced
40 by increasing the number of divisions during the search. In this work we pro-
41 pose a new algorithm specially designed for continuous multimodal optimisation,
42 Region-based Memetic Algorithm with Archive (RMAwA). Although RMAwA
43 maintains the same definition of a niche and alternatively applies the EA and
44 the LS, the memetic scheme is modified and a novel archive is implemented to
45 match the requirements of multimodal optimisation. First, while RMA-LSCh
46 uses LS Chaining [32, 33] and thus limites the number of fitness evaluation per
47 LS application, RMAwA applies the LS until it has reached a local or global
48 optimum. Most importantly, regions intensively explored by LS are discarded
49 by the proposal from further exploration. RMAwA contains an indexed archive
50 with these regions to reduce the search domain in a very efficient way. Also,
51 because the identified optima are stored into the archive and not into the pop-

52 ulation, the number of optima that RMAwA can identify is not limited by the
53 population size [12, 63, 64].

54 RMAwA is tested using a specific benchmark for multimodal optimisation.
55 The experiments carried out show that the use of the region based niching strat-
56 egy coupled with an archive provides interesting improvements to the memetic
57 framework, and that the RMAwA is a very competitive algorithm against ex-
58 isting ones.

59 This paper is organised as follows. In Section 2, we present a quick intro-
60 duction on methods previously proposed to tackle multimodal problem optimi-
61 sation. In Section 3, we present the RMAwA and detail each component. In
62 Section 4, we explain the experimental framework used and the parameter set-
63 ting of the algorithm. In Section 5, several comparisons are carried out to study
64 the influence of the different components of the algorithm and our proposal is
65 compared with other algorithms in the literature. Finally, In Section 6 some
66 concluding remarks are pointed out.

67 2. Background

68 In order to identify multiple optima of a fitness landscape several techniques
69 have been proposed. In this section, we give a brief overview of techniques that
70 have been proposed to maintain the diversity in the population in order to pre-
71 vent its convergence towards a single optimum. Such techniques are commonly
72 called niching strategies and refer to the technique used for the discovery and
73 preservation of distinct niches. This term is a reference to the ecological concept
74 of niches referring to the formation of distinct species exploiting different niches
75 (resources) in an ecosystem.

76 The main challenge in multimodal optimisation is the unknown nature and
77 characteristics of the objective function, specifically the number of global optima
78 and their repartition on the search domain. The main goal of the proposals
79 presented in this section is to tackle these issues. Alternatively, [55] proposes a
80 preprocessing tool to estimate the number of basins of attraction in the fitness
81 landscape.

82 We have classified the methods proposed to tackle multimodal optimisation
83 into two categories. The first one lists the classical niching strategies which
84 mainly affect the replacement criterion of the EA they are applied to. The
85 second one works with the idea of creating subgroups of solutions in different
86 area of the search space by limiting the cooperation of each individual to its
87 nearest neighbours. We refer to them as neighbourhood based techniques.

88 In this section, we first describe the different elements composing those two
89 categories by giving a general overview of the proposal making use of such
90 techniques. In a third section, we briefly introduce proposals combining those
91 techniques with MA which demonstrate that the use of a refinement method
92 improves the performance of EAs for multimodal optimisation.

93 *2.1. Classical niching techniques*

94 The first niching techniques consist in limiting the presence of multiple solu-
 95 tions within the same niche in order to keep the population highly diverse.
 96 When included in a classical EA, those mechanisms are mainly replacement
 97 strategies designed to remove solutions present in the same vicinity. We de-
 98 scribe here the four main methods to achieve this objective: crowding, clearing,
 99 fitness sharing, and speciation.

100 *2.1.1. Crowding*

101 Crowding is one of the first techniques proposed to tackle multimodal opti-
 102 misation problems [7]. After the generation of a new solution, a random sample
 103 of CF solutions is selected in the population. Each new solution competes with
 104 the closest solution of the sample to stay in the population. This technique's
 105 main drawback is the definition of the crowding factor parameter (CF). A small
 106 value can lead to the replacement of a distant solution to the offspring and thus
 107 a loss of information, and a very large value has a high computational cost. The
 108 efficiency of this technique has proven to be limited [30] and advanced versions
 109 have been proposed:

- 110 • *Deterministic crowding* proposed by [30] tries to limit the problem of
 111 replacement errors induced by the crowding technique by eliminating the
 112 need of defining the CF parameter. To do so, an offspring competes with
 113 its own parents to stay in the population.
- 114 • *Probabilistic crowding* [31] on the other hand modifies the replacement
 115 strategy of the original technique. In this scheme, the offspring and its
 116 most similar individual in the crowding sample compete in a probabilistic
 117 tournament where the probabilities of winning for each individual X ,
 118 $p(X)$, is calculated according to their fitness:

$$p(X) = \frac{f(X)}{f(X) + f(Y)} \quad (1)$$

119 where $f(X)$ is the fitness of the same solution X and $f(Y)$ is the fitness of
 120 the other solution. The idea is not to always show preference to solutions
 121 with higher fitnesses which may lead to the loss of niches.

122 In [57], Thomsen proposed the popular crowding differential evolution (CDE)
 123 applying a classical crowding strategy on a differential evolution (DE) where a
 124 new solution is created by means of classical DE mutation and crossover scheme
 125 comparing with its closest solution in the whole population for replacement.

126 CDE was then extended to multi-population crowding DE (MCDE) in [63]
 127 where multiple sub-population evolve in parallel using CDE. When all the sub-
 128 populations have converged, the optima identified by each of them are stored in
 129 an archive and the sub-populations are reinitialised.

130 More recently, Qu et al. proposed the dynamic grouping of CDE (DGCDE)
 131 [45] with ensemble of parameters. The population is divided into three sub-
 132 population to which a set of control parameters is assigned.

133 In [44], Qing et al. proposed a Crowding Clustering Genetic Algorithm
 134 (CCGA) using a clustering technique to eliminate the genetic drift introduced
 135 by the crowding strategy.

136 2.1.2. Clearing

137 Clearing techniques [42] lie in the principle of dedicating the limited re-
 138 sources of a niche to its best individuals. The population is sorted according
 139 to the individual fitness values. The solutions are then selected one after the
 140 other and the solutions with worse fitness falling within their niche radius σ_{clear}
 141 are removed. Clearing has a low complexity and shows the best performances
 142 amongst the classical techniques but is highly sensitive to the niche radius [51].

143 Variations have then been proposed to limit influence of the σ_{clear} parameter.
 144 For instance, in [47], similarly to the previously cited DGCDE, the authors
 145 propose an ensemble of clearing DE (ECLDE) in which the population was
 146 equally divided into 3 sub-populations each evolving in parallel using a clearing
 147 DE with different values of σ_{clear} .

148 Some techniques use a redefinition of the niche in order to remove the use
 149 of the parameter σ_{clear} . In [11], the niches are defined through a hill-valley
 150 detection mechanism instead of using a niche radius. In [50], the niches are
 151 defined by fuzzy clustering of the solutions of the populations.

152 2.1.3. Fitness sharing

Contrarily to the clearing technique which consist in dedicating niche re-
 sources to a single solution, fitness sharing [16] consists in reducing the fitness
 of individuals present in densely populated regions. The fitness used of the i th
 individual, $f_{shared}(i)$, is calculated by:

$$f_{shared}(i) = \frac{f_{original}(i)}{\sum_{j=1}^{NP} sh(d_{ij})} \quad (2)$$

where $f_{original}$ is the original fitness function, NP is the population size,
 and sh function is calculated by:

$$sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}}\right)^\alpha, & \text{if } d_{ij} < \sigma_{share} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

153 where d_{ij} is the distance between individual i and j , σ_{share} is the sharing
 154 radius and α is a constant called sharing level.

155 In [57], Thomasen also proposed an DE using sharing where, after each gen-
 156 eration, the new shared fitnesses are calculated over the population individuals
 157 and the trial vectors, the best half being kept in the population.

158 *2.1.4. Speciation*

159 Proposed in [22], speciation or species conservation introduces the notion of
 160 species by separating the population into several groups (species) according to
 161 their similarity. Those species are identified by a dominating individual called
 162 the species seed and a species distance $\sigma_{species}$ defining the maximum distance
 163 between two individual of the same species. The set of species seed is build at
 164 each generation by iteratively adding individuals from the population that are
 165 further from any species seed than $\sigma_{species}/2$. The individuals are kept from one
 166 generation to another until a better solution is identified within their species
 167 while the classical recombination operators are applied.

168 In [23], this concept is applied to a speciation-based PSO (SPSO). In SPSO,
 169 the particles are gathered into species to form sub-populations. This proposal
 170 was later extended to reduce its dependency to the species distance parameter
 171 by using population statistics [3] and a time-based convergence measure [49].

172 *2.2. Neighborhood based technique*

173 Another class of niching strategies can be referred to as neighbourhood-
 174 based. Contrarily to the previous section where the niching strategy could be
 175 seen as replacement strategy, these methods use the geographical information
 176 of the solutions in a population to modify the recombination scheme of a given
 177 EA. The main idea is to make solutions by only considering their neighbours in
 178 order to emphasize the speciation.

179 Originally named spatially-structured EAs (SSEA) [58], these algorithms
 180 form sub-populations of individuals (called deme) based on their similarity and
 181 perform genetic operations within each deme.

182 This idea has then been extended and two kinds of neighbourhoods can be
 183 identified in the literature:

- 184 • *Index-based neighbourhood* [24] uses the indices in the population of a PSO
 185 to identify the neighbourhood of a solution. The velocity of a particle is
 186 thus influenced by the local best solution instead of the global best.
- 187 • *Distance-based neighbourhood* uses the euclidean distance between individ-
 188 uals. In [26], the author proposed the FER-PSO algorithm where parti-
 189 cles are attracted towards the "fittest-and-closest" neighbours. Similarly,
 190 the notion of neighbourhood is applied for DE in [13]. A new mutation
 191 strategy, DE/nrand/x is proposed. It uses as a base vector the nearest
 192 neighbour of each individual. This mutation strategy has then been used
 193 for more advanced models like in [12]. In [4] a neighbourhood mutation
 194 is proposed that considers normalized distance. Another option is to use
 195 the distance to create a clustering partition of the population to maintain
 196 diversity [15].

197 Neighbourhood-based strategies have often been coupled with classical nich-
 198 ing strategies. For instance in [10], the authors propose including in a SSEA a
 199 fitness sharing and a clearing strategy.

200 In [48], the authors use the DE/nrand/x operator with crowding, sharing
 201 and species-based niching strategies and obtain better results than the original
 202 algorithms.

203 *2.3. Memetic algorithms for multimodal optimisation*

204 As stated in the introduction, MA are the hybridisation of an EA and a
 205 LS method. This model is part of the more general Memetic computing (MC)
 206 family of algorithms which combine various optimisers (memes). The efficiency
 207 of these models have helped them gain popularity over the past decade [5, 37].

208 The coordination of the memes is the main research topic in MC. Ong et al.
 209 [40] proposed a classification which was later updated by Neri et al. [37]:

- 210 • Adaptive Hyper-heuristic [19]: the memes are coordinated by means of
 211 heuristic rules.
- 212 • Meta-Lamarckian learning [39]: the probabilities of using the memes are
 213 based on their success, providing an online adaptability.
- 214 • Self-Adaptive and Co-Evolutionary [20, 54]: the memes are encoded with
 215 the candidate solutions and evolve in parallel so the most appropriate can
 216 be selected.
- 217 • Fitness Diversity-Adaptive [38]: the selection of the memes to be operated
 218 is based on the diversity measure of the population.

219 MA are particularly adapted to multimodal optimisation problems as, when
 220 applied to different solutions, an LS method can offer a strong refinement of
 221 the promising solutions discovered by the EA, providing great accuracy for the
 222 identification of multiple optima. The use of such model has raised interest in
 223 the research community.

224 For instance, the Sequential Niching Memetic Algorithm (SNMA) proposed
 225 by Vitela et al. in [60] and then extended in [61] is an MA which combines a ge-
 226 netic algorithm (GA) with a gradient-based LS method. Before each generation,
 227 the LS is applied to each solution of the population.

228 In [46], Qu et al. included an LS method to various previously cited PSO for
 229 multimodal optimisation (FER-PSO, SPSO, rPSO). The LS method used con-
 230 sisted in generating at each iteration new solutions in the neighbourhood of the
 231 personal best of each particle to explore its surrounding. They demonstrated
 232 that the resulting memetic PSO obtained better results than the original algo-
 233 rithms. Similarly, Wang et al. proposed a memetic SPSO [62] which adaptively
 234 uses two different LS methods and came to the same conclusions.

235 **3. Region-based memetic algorithm with archive**

236 In this section we present the region-based MA with archive (RMAwA), an
 237 algorithm designed for multimodal optimisation which uses a niching technique
 238 to obtain as much optima as possible.

239 RMAwA is a MA which alternatively applies an EA through a certain num-
 240 ber of evaluations and a LS method to the best solution in the population until
 241 stagnation. It then considers that an optimum has been reached, thus it stores
 242 that solution in an external archive and the EA is carried on.

243 To maintain diversity during the search the algorithm divides each dimension
 244 in regions of same size, dividing the domain search in hypercubes. RMAwA uses
 245 these regions in two ways: First, only one solution is allowed in each region,
 246 thus when a solution generated by the EA falls in a region already occupied by
 247 a solution of the population the worst is removed. Second, regions in which one
 248 optimum has been found, by means of LS, are considered to be explored enough
 249 and discarded from the search space. The size of regions decreases during the
 250 run, by increasing the number of divisions per dimension.

251 In order to efficiently discard regions from further exploration, this model
 252 maintains an index of the regions represented by a solution in the archive. Also,
 253 it stores all the found optima to recalculate the regions when its number changes.

254 In the following subsections, we detail the algorithm. First, we briefly de-
 255 scribe the concept of the region-based niching strategy. Then, we explain the
 256 general scheme of the algorithm along with how the different components are
 257 integrated. Finally, we explain how the archive works in detail: its structure,
 258 which solutions are stored, and how it is used.

259 3.1. Region-based niching strategy

260 In [21], a novel niching strategy was proposed that redefines the notion
 261 of niche from the area surrounding each solution in the population to a fixed
 262 division of the search space. Each dimension of the search space is divided into a
 263 certain number of divisions, ND , creating a predefined grid of equal hypercubes
 264 representing the niches.

265 In [56, 52], the authors use a similar partitioning of the search space to
 266 approximate the basin of attractions in multimodal fitness landscapes by means
 267 of clustered genetic search. In our algorithm, this fragmentation is used to define
 268 different niches in the search space. Ideally, regions contain a single basin of
 269 attraction but the unpredictability of the number of optima and their repartition
 270 in the search domain can not guarantee that. An illustration of the divisions of
 271 the search space can be seen in Figure 1. A solution $s_n \in \mathbb{R}^D$ is a real-parameter
 272 vector representing a solution to the problem at hand. It is associated with its
 273 region identified by its indices in each dimension, represented by a vector of
 274 integer values $r_n \in \mathbb{N}^D$. The advantage of such definition is to allow faster
 275 retrieval of the existing niches by avoiding the computationally expensive cost
 276 of calculating the euclidean distance between solutions.

277 In a region-based niching strategy, solutions generated in the evolutionary
 278 process compete with either the current solution present in the same region or
 279 the worst individual of the population. This technique can thus be assimilated
 280 to a clearing strategy in the sense that solutions compete to represent each niche
 281 in the population. The difference with classical niching strategy is the definition
 282 of the niche going from an euclidean distance-based representation to a region-
 283 based representation. In order to reduce the influence of the niche/region size, a

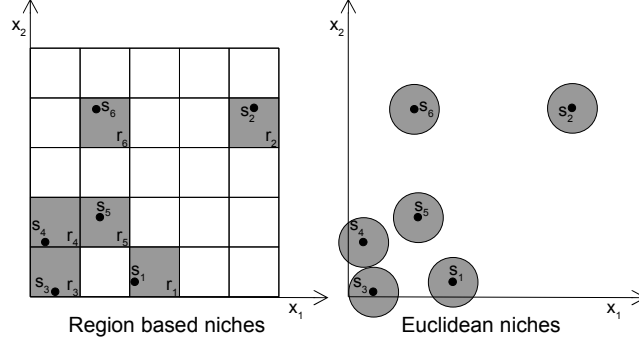


Figure 1: Example of region-based niches and distance-based niches

284 commonly critical parameter in niching strategies, following the idea proposed
 285 in [21], the region size is decreased along the search, as it is detailed in the
 286 following subsection.

287 3.2. General Scheme

288 Considering the classification described in Section 2.3, RMAwA uses an
 289 adaptive hyper-heuristic strategy. It alternatively applies an EA and a LS
 290 method. The EA is applied over the population during I_{EA} evaluations and
 291 then the best solution of the population s_{best} is selected for local improvements
 292 by the LS until the LS cannot bring about any other significant improvement.
 293 This loop is repeated until the given maximum number of evaluations Max_{FEs}
 294 is reached. The general scheme of the algorithm can be seen in Algorithm 1.

Algorithm 1 Pseudo-code for general scheme of the RMAwA

- 1: Initialise population with uniform distribution over the whole search space
 - 2: **while** Max_{FEs} is not reached **do**
 - 3: Apply SSGA with i_{EA} evaluations following Algorithm 2
 - 4: Select the best individual in the population s_{best}
 - 5: Apply LS method following Algorithm 3 on s_{best}
 - 6: **if** conditions for number of divisions update **then**
 - 7: Update number of divisions: $ND_i = m_u \cdot ND_{i-1}$
 - 8: Update index of the archive
 - 9: **end if**
 - 10: **end while**
-

295 In the proposal, when the EA generates a solution in a region with a existing
 296 solution, the worst is removed. By increasing the number of regions, we also try
 297 to reduce the possibility for the EA to encounter more and more difficulties in
 298 finding new solutions falling in regions not already represented in the archive.

299 With the region definition of a niche, the region size is defined by the number
 300 of divisions per dimension ND . We consider that the stopping criterion is a

301 predefined maximum number of fitness evaluations Max_{FEs} . ND starts with
 302 a initial value ND_0 . Then, ND is increased u times throughout the search by
 303 $ND_i = m_u \cdot ND_{i-1}$ where m_u is the multiplier of the number of division. An
 304 update occurs every $Max_{FEs}/(u+1)$. The values for parameters ND_0 , u and
 305 m_u are indicated in Section 4. In order to prevent the search from stalling,
 306 an update of ND also occurs if every region has been explored by the LS and
 307 are represented in the archive. This situation is very likely to happen in low
 308 dimensionality. For each update, the corresponding regions of each solution in
 309 the population are recalculated and the archive updates the regions according
 310 to the solutions presented.

311 The following two sections describe the EA and the LS method used and
 312 how they are incorporated in the RMAWA.

313 3.3. The EA

314 The EA in RMAWA evolves a population of solutions over the whole search
 315 space seeking promising solutions for the LS method to refine. The evolution
 316 process is orientated by the region-based niching strategy and the set of excluded
 317 regions from the archive.

Algorithm 2 Pseudo-code for the EA in RMAWA

```

1:  $i = 0$ 
2: while  $i < i_{EA}$  do
3:   Select two parents in the population
4:   repeat
5:     Create an offspring  $s_n$  using crossover and mutation
6:     Calculate the region  $r_n$  to which  $s_n$  belongs
7:   until  $r_n$  is not represented in the archive
8:   Evaluate  $s_n$ ,  $i = i + 1$ 
9:   Retrieve from the population the set of solutions  $S_{r_n}$  of solutions belong-
   ing to the region  $r_n$ 
10:  if  $S_{r_n} \neq \emptyset$  then
11:    set  $S_{r_n} = S_{r_n} \cup s_n$ 
12:    Remove worst individual from  $S_{r_n}$ 
13:  else
14:    Replace the worst individual  $s_{worst}$  in the population if  $f(s_n)$  is better
   than  $f(s_{worst})$ 
15:  end if
16: end while

```

318 The EA used here, as in the RMA-LSCh, is a steady-state genetic algorithm
 319 (SSGA). On each application, the algorithm runs over i_{EA} evaluations. Two
 320 parents are selected by means of *negative assortative mating* strategy (NAM)
 321 [1] (with a pool size of 3). Offspring are generated using a BLX- α crossover
 322 operator [14] and the BGA mutation operator [36]. The EA in the RMAWA is
 323 described in Algorithm 2.

324 When a new solution s_n is generated via the operators described above,
 325 it goes through different processes before validation. First, the region r_n it
 326 belongs to is calculated. Then, r_n is looked for in the archive. If this region is
 327 already represented by one optimum in the archive, s_n is discarded and thus not
 328 evaluated. Otherwise, s_n is evaluated and compared with the set of solutions
 329 from the population present in the same region r_n . The worst solution is then
 330 removed and replaced by s_n . If r_n is not yet represented in the population, then
 331 s_n competes with the worst solution of the whole population to replace it.

332 3.4. The LS method

333 The continuous LS algorithm used here is CMA-ES [17]. This algorithm
 334 is the *state-of-the-art* in continuous optimisation. Thanks to the adaptability
 335 of its parameters, its convergence is very fast and obtains very good results.
 336 CMA-ES uses a distribution function to obtain new solutions, and adapts the
 337 distribution around the best created solutions.

338 Contrarily to RMA-LSCh, RMAwA does not implement a LS chaining mech-
 339 anism because the local search here is applied to the same solution until it cannot
 340 be improved anymore. This modification is due to the fact that this algorithm
 341 considers as optima solutions those which cannot be improved by LS application.

342 As stated before, the best solution s_{best} of the population is selected for local
 343 refinement. To ensure that this solution will not take part in further exploration,
 344 it is removed from the population, placed in the archive and replaced by a
 345 random solution. The LS is applied multiple times with i_{LS} evaluations until
 346 the last application does not bring about any other sufficient improvement.
 347 Between each application, the parameters of the previous LS application are
 348 retrieved to carry on from the point where it stopped. In the case of CMA-ES,
 349 the learnt covariance matrix is thus reused from one application to another.
 350 The final solution is then stored in the archive. The application of the LS is
 351 described in Algorithm 3.

Algorithm 3 Pseudo-code for the application of the LS in RMAwA

- 1: Add s_{best} to the archive
 - 2: $s_{LS}^0 = s_{best}$
 - 3: Replace s_{best} by a random solution in the population
 - 4: **repeat**
 - 5: Apply the LS method to s_{LS}^t with i_{LS} evaluations, giving s_{LS}^{t+1}
 - 6: **until** $|f(s_{LS}^t) - f(s_{LS}^{t+1})| < \delta_{LS}^{min}$
 - 7: Add s_{LS}^t to archive
-

352 3.5. The archive

353 As described previously, this algorithm implements an archive aiming at
 354 storing solutions considered as optimised (solutions that have been refined by
 355 the LS method) and creating an index of regions of the search space considered
 356 undesirable for further exploration.

357 We describe in this section the structure of the archive allowing such mech-
 358 anisms. We then characterise the solutions which are inserted in the archive to
 359 define their region as undesirable.

360 3.5.1. Structure

361 The archive is composed of two collections and its size is not limited. The
 362 first one is a simple list of real-value solutions that store the detected optima.
 363 The second one is a sorted index of the regions represented by the solutions in
 364 the previous list. The regions listed in the index are considered as forbidden
 365 areas for the generation of future solutions by the EA. The index is a self-
 366 balancing binary search tree which offers an insertion and search complexity of
 367 $O(\log n)$. This low complexity allows a large amount of solutions to be stored in
 368 the archive with a limited computational cost. Moreover, it only allows unique
 369 elements to be stored.

370 In Figure 2, we show an example of the archive structures in the continuity
 371 of the representation of the search space in Figure 1. We can see how a new
 372 solution, composed by the actual real-value solution s_n and the indices of the
 373 region it belongs to r_n , are used. The former is stored in the archive while the
 374 latter is added to the index. If a region is represented by multiple solutions
 375 in the archive, there will be only one entry in the index for that region. The
 376 following section describes what regions are considered as restricted to further
 377 exploration.

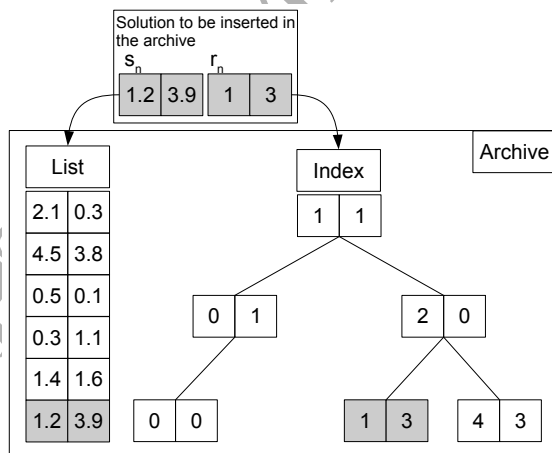


Figure 2: Example of the representation of the archive and its index for two-dimensional problems

378 3.5.2. Solutions stored the archive

379 The main purpose of the archive is to store optima identified during the
 380 search. Knowing when an optima is found can however be complicated if the

381 fitness value of the optima is unknown. Thanks to the use of an LS method, we
 382 consider a solution as an optimum (local or global) when the last LS application
 383 does not bring sufficient improvement. Insufficient improvement occurs when
 384 the difference between the fitness of the starting point of the LS and the fitness
 385 of the obtained solution is below δ_{LS}^{min} .

386 Apart from storing the optima found by means of LS, the archive also saves
 387 the solution that serves as the starting point of each LS application. The idea
 388 behind this is to also eliminate from the search space regions that lead to already
 389 identified optima.

390 To summarise, the archive stores the solutions that have undergone LS ap-
 391 plications. The rationale behind this is to ensure that the regions in the archive
 392 and thus removed from the search space have been intensively explored. How-
 393 ever, depending on the characteristics of the fitness landscape, it is not guar-
 394 anteed that several optima are not in the same region. This risk is reduced by
 395 decreasing the niche size during the search as is described above.

396 3.5.3. Updating the niche size

397 The update of the number of divisions per dimensions, i.e. the niche size,
 398 is performed in order to prevent the presence of multiple optima in the same
 399 region. This process is particularly important in this model as some of the
 400 regions are completely discarded from the search which may lead to ignoring
 401 a number of optima. When an update is performed, as the regions indices are
 402 modified and the archive index is wiped:

- 403 • A new index is created from the resulting list of regions. Because the
 404 solutions are kept in the population, its corresponding regions (using the
 405 new size) are calculated again and stored in the archive. The number of
 406 stored regions is maintained but the indexes make reference to smaller
 407 regions.
- 408 • The regions of the solutions stored in the archive are recalculated according
 409 to the new partitioning of the search space.

410 In summary, the archive has to be recalculated with each update of the
 411 niche size, thus its structure is designed to carry out the operation easily and
 412 efficiently.

413 4. Experimental framework

414 The experiments in this paper were carried out using the benchmark pro-
 415 posed for the special session and competition on niching methods for multimodal
 416 function optimisation of the IEEE Congress on Evolutionary Computation in
 417 2013 (CEC'2013) [25]. In this section, we describe the framework used to per-
 418 form these experiments: first we describe the benchmark used and the evaluation
 419 method, and then we explain the parameter tuning used for the final version of
 420 the algorithm.

421 *4.1. The CEC'2013 benchmark*

422 The CEC'2013 benchmark offers a set of continuous objective functions $f : \mathcal{D} \rightarrow \mathbb{R}$ where $\mathcal{D} \subset \mathbb{R}^D$ defines the bounded subset of \mathbb{R}^D . The objective consists
 423 in identifying every $x \in \mathcal{D}$ such that $x = \operatorname{argmin}_{z \in \mathcal{D}} \{f(z)\}$. Functions in this
 424 benchmark are to be tackled as black-box problems, i.e. the use of differential
 425 based methods is not allowed. Each function contains a finite number of global
 426 of optima.

427 The CEC'2013 benchmark is composed of 12 bounded functions :

- 428 • f_1 : Five-Uneven-Peak Trap, $f_1(x)$ where $x \in [0, 30]$, $D = 1$
- 429 • f_2 : Equal Maxima, $f_2(x)$ where $x \in [0, 1]$, $D = 1$
- 430 • f_3 : Uneven Decreasing Maxima, $f_3(x)$ where $x \in [0, 1]$, $D = 1$
- 431 • f_4 : Himmelblau, $f_4(\vec{x})$ where $\vec{x} \in [-6, 6]^D$, $D = 2$
- 432 • f_5 : Six-Hump Camel Back, $f_5(x_1, x_2)$ where $x_1 \in [-1.9, 1.9]$ and $x_2 \in [-1.1, 1.1]$, $D = 2$
- 433 • f_6 : Shubert, $f_6(\vec{x})$ where $\vec{x} \in [-10, 10]^D$, $D = \{2, 3\}$
- 434 • f_7 : Vincent, $f_7(\vec{x})$ where $\vec{x} \in [0.25, 10]^D$, $D = \{2, 3\}$
- 435 • f_8 : Modified Rastrigin - All Global Optima, $f_8(\vec{x})$ where $\vec{x} \in [0, 1]^D$,
 436 $D = 2$
- 437 • f_9 : Composition Function 1, $f_9(\vec{x})$ where $\vec{x} \in [-5, 5]^D$, $D = 2$
- 438 • f_{10} : Composition Function 2, $f_{10}(\vec{x})$ where $\vec{x} \in [-5, 5]^D$, $D = 2$
- 439 • f_{11} : Composition Function 3, $f_{11}(\vec{x})$ where $\vec{x} \in [-5, 5]^D$, $D = \{2, 3, 5, 10\}$
- 440 • f_{12} : Composition Function 4, $f_{12}(\vec{x})$ where $\vec{x} \in [-5, 5]^D$, $D = \{3, 5, 10, 20\}$

441 Some function are presented with different dimensionality creating a total of
 442 20 problems. Table 1 details the 20 problems and their characteristics. In this
 443 paper, we refer by f_i to i -th function and F_j to the j -th problem, a problem
 444 consisting of the pair $\{f_i, D\}$ where D is the dimensionality of the problem.
 445 We are only interested here in identifying the global optima. The number of
 446 global optima is known and finite, but this information cannot be used in the
 447 optimisation process. More details on each function can be seen in [25].

448 *4.2. Evaluation*

449 For the evaluation of an algorithm's performance over multiple run (50 runs
 450 to be executed following the competition requirements), we use the now com-
 451 monly used *peak ratio* (PR). The PR is the average percentage of found optima
 452
 453

Table 1: CEC'2013 benchmark problems

Problem	Function	D	Number of optima	Max_{FEs}
F_1	f_1	1	2	$5 \cdot 10^4$
F_2	f_2	1	5	$5 \cdot 10^4$
F_3	f_3	1	1	$5 \cdot 10^4$
F_4	f_4	2	4	$5 \cdot 10^4$
F_5	f_5	2	2	$5 \cdot 10^4$
F_6	f_6	2	18	$2 \cdot 10^5$
F_7	f_7	2	36	$2 \cdot 10^5$
F_8	f_6	3	81	$4 \cdot 10^5$
F_9	f_7	3	216	$4 \cdot 10^5$
F_{10}	f_8	2	12	$2 \cdot 10^5$
F_{11}	f_9	2	6	$2 \cdot 10^5$
F_{12}	f_{10}	2	8	$2 \cdot 10^5$
F_{13}	f_{11}	2	6	$2 \cdot 10^5$
F_{14}	f_{11}	3	6	$4 \cdot 10^5$
F_{15}	f_{12}	3	8	$4 \cdot 10^5$
F_{16}	f_{11}	5	6	$4 \cdot 10^5$
F_{17}	f_{12}	5	8	$4 \cdot 10^5$
F_{18}	f_{11}	10	6	$4 \cdot 10^5$
F_{19}	f_{12}	10	6	$4 \cdot 10^5$
F_{20}	f_{12}	20	8	$4 \cdot 10^5$

454 over all global optima within the Max_{FEs} evaluations, and it is calculated by
 455 following Eq. 4:

$$PR = \frac{\sum_{i=1}^{NR} NPF_i}{NKP * NR} \quad (4)$$

456 where NPF_i is the number of global optima found in the i th run, NKP is
 457 the number of known global optima and NR is the number of runs (for this
 458 benchmark, $NR = 50$). It is considered that an optimum $optim$ is obtained if
 459 a solution sol is found where $dist(sol, optim) \leq \epsilon$, where dim is the Euclidean
 460 distance, and ϵ is a real value called accuracy level. The PR are calculated
 461 according to five different accuracy levels $\epsilon = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$.

462 Comparisons between algorithms have been performed for each accuracy
 463 level independently. For the comparison of two algorithms we considered non-
 464 parametric statistical tests [9]. More specifically, we used the Wilcoxon matched-
 465 pairs signed ranks tests for the direct comparison of two algorithms.

466 4.3. Automatic configuration

467 Setting the parameters of a new proposal can be a long and tedious task.
 468 Moreover, it does not ensure an optimal setting for these parameters. Consider-
 469 ing the novelty of certain components in this algorithm, it is more reliable to use
 470 an automatic configuration tool to assist in the design of the algorithm tuning
 471 the most critical parameters. To do so, we have used IRACE [29]. The IRACE
 472 package has already been extensively tested in several research projects, leading

473 to successful improvement of the state-of-the-art, see for instance [28, 27]. The
 474 reader may refer to [41] for more information about IRACE and its parameters
 475 (we have used the recommended parameter values).

476 We selected a set of parameters that we considered the most critical, and
 477 tuned them over the 20 problems of the CEC'2013 benchmark. For the non-
 478 tuned parameters we have selected commonly used values when not recom-
 479 mended values where given by from its authors. The list of tuned parameters
 480 can be seen in Table 2, showing for each parameter the explored range and the
 481 final value obtained by IRACE.

Table 2: Tuned parameters and obtained values

Parameters	Descriptions	Ranges	Tuned
i_{EA}	EA intensity, number of evaluations allocated to each EA application	[100, 1000]	550
i_{LS}	LS intensity, number of evaluations allocated to each LS application	[100, 1000]	150
ND_0	Initial number of divisions, defines the size of the niches/regions	[2, 10]	2
u	Number of update to be performed	[2, 5]	4
m_u	Update multiplier	[1, 5]	1.7
NP	Population size of the EA	[40, 120]	70
α	Parameter for the $BLX - \alpha$ crossover	[0.1, 0.9]	0.9

482 We can note that the EA intensity is almost four times the LS intensity.
 483 This is due to the fact that the LS is applied multiple times (until the improve-
 484 ments brought not significant enough) in each cycle. Concerning the number of
 485 division, we can see that the smallest number of divisions have been preferred
 486 ($ND_0 = 2$) along with a slow increase during the search by multiplying four
 487 times by 1.7: $ND_{i+1} = \text{ceil}(1.7 \cdot ND_i)$. The number of the divisions sequence
 488 is then [2, 4, 7, 12, 21]. Finally an important thing to note is the value of the α
 489 parameter for the $BLX - \alpha$. Set to a high value ($\alpha = 0.9$), it gives the EA a great
 490 exploration range.

491 The other parameters listed in Table 3 were left to their default values taken
 492 from the corresponding papers. δ_{LS}^{min} defines the accuracy required for the search
 493 and is set to 10^{-6} as the highest accuracy level required is 10^{-5} . Concerning
 494 CMA-ES problems, we have set them to the default values as given in [17]. and
 495 the size of NAM selection method is taken from the previous work in [21].

496 The parameters presented in Table 2 and 3 are the ones used in every ex-
 497 periment performed on every function and dimension of the benchmark.

Table 3: Other parameters

Parameters	Descriptions	Value
λ	Parameter to define the CMA-ES population size $p = 4 + \lambda \ln(D)$	3 [17]
μ	Defines the parent size for the CMA-ES p/μ	2 [17]
NAM_{size}	Size of the NAM selection method	3 [21]
δ_{LS}^{min}	Threshold for the LS stopping criterion	10^{-6}

4.4. Possibility of finding all optima

In this section, we discuss the ability of RMAwA to find all optima with an unlimited number evaluation. In other words, we wish to ensure that the search is not restricted to any subset of the whole search domain. For this model, we identify two phenomena that can cause such restriction and we discuss here if their occurrence is possible in the proposal.

First, in population-based algorithm the risk of premature convergence of the population may lead to a genetic drift. The fact that RMAwA regularly generates new random solutions (when a solution is placed in the archive, it is replaced by a random solution) ensures sufficient diversity in the population to prevent premature convergence.

The second risk that can be identified in this model is due to the restriction of the search to regions represented in the archive. Indeed, if a region represented in the archive contains more than one optimum, some optima might be ignored. The probability of having more than one optimum present in the same region (noted M) is directly proportional to the hyper-volume of the regions V_r calculated by Eq. 5:

$$P(M) = a.V_r \quad (5)$$

where a is a variable that is dependent on the objective function f and the search domain. Basically, the smaller the region, the less probable that it will contain multiple optima. Thanks to the region size update, V_r keeps decreasing during the search. In our algorithm, we make a limited number of reductions because the fitness evaluation number is also very limited. For an extremely large fitness evaluation number, the reductions would be applied repeatedly, reducing the hyper-volume of the regions each time. Thus, for an unlimited number of evaluations Max_{FEs} :

$$\lim_{Max_{FEs} \rightarrow +\infty} V_r = 0 \quad (6)$$

523 Hence:

$$\lim_{Max_{FEs} \rightarrow +\infty} P(M) = 0 \quad (7)$$

524 Thus, there is no risk of limiting the search.

525 5. Experimental results

526 In this section, we are going to study the behaviour of the different compo-
527 nents of our proposal, and we are going to compare our algorithm to previous
528 algorithms in the literature. All the experiments are carried out following the
529 experimental framework explained in previous section.

530 The analysis of our proposal include the following experiments: First, we
531 prove that using the region definition of a niche compared to the euclidean def-
532 inition is more efficient in terms of computational time and exploration. Then,
533 we demonstrate that using the solutions in the archive as excluding regions en-
534 hance the performance of the model. We also analyse the influence of the region
535 based niching strategy with the archive on the diversity of the population and in
536 the exploration factor. Then, we analyse the memory and computational cost of
537 the archive and the different components of the algorithm. Finally, we compare
538 the proposed algorithm RMAwA with existing algorithms.

539 5.1. Region niches versus classical niches

540 Here, we assess the efficiency in terms of computation time and performance
541 of the region definition of niches against the classical definition which implies
542 calculating the euclidean distance between solutions. To do so, we consider the
543 model presented without the use of the archive.

544 The resulting algorithm here simply referred to as region based memetic
545 algorithm (Region-MA) is opposed to an equivalent algorithm which uses the
546 euclidean distance based definition of a niche as it is used in the classical clear-
547 ing algorithm. This version is referred to as euclidean-distance based memetic
548 algorithm (Euclidean-MA). On the generation of a new solution by the EA, the
549 offspring created compete with the solutions falling within its niche radius σ ,
550 which is set to half the size of a region. In Region-MA, as it is explained in
551 Section 3.1, new solutions created by the EA compete with the solutions already
552 in the same regions.

553 In order to simplify the display of the results, we will only focus on the
554 highest level of accuracy ($\epsilon = 10^{-5}$). Indeed, the definition of a niche only
555 affects the ability of the algorithm to explore the search space and not the
556 precision of the solutions obtained.

557 In Table 4, we show the PRs obtained by both versions along with the exe-
558 cution time difference in percentage. We can see that the results of Region-MA
559 are clearly better, and with significant differences (comparing with Wilcoxon's
560 test, the use of regions is statistically better with a p-value < 0.001 , see Table 5).
561 Also, the execution time is much smaller, over the whole benchmark, using the
562 region-based niches saves up to 17.4% of time.

Table 4: PRs (for $\epsilon = 10^{-5}$) obtained by Region-MA and Euclidean-MA and execution time difference (in percentage)

Problem	F_1	F_2	F_3	F_4	F_5
Region-MA	0.81	0.42	1	0.97	0.99
Euclidean-MA	0.77	0.56	1	0.36	0.87
Time difference (%)	-35.88	-26.10	-28.36	-45.20	-43.57
Problem	F_6	F_7	F_8	F_9	F_{10}
Region-MA	0	0.7	0.06	0.22	0.94
Euclidean-MA	0	0.05	0.06	0.01	0.13
Time difference (%)	-30.26	-39.05	-42.13	-38.96	-24.89
Problem	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
Region-MA	0.68	0.86	0.63	0.64	0.15
Euclidean-MA	0.27	0.14	0.2	0.18	0.14
Time difference (%)	-19.42	-20.90	-28.38	-19.20	-21.11
Problem	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}
Region-MA	0.36	0.16	0.17	0.13	0.13
Euclidean-MA	0.19	0.13	0.17	0.13	0.13
Time difference (%)	-15.93	-1.56	-25.74	-21.19	-7.42

Table 5: Wilcoxon comparison of the PR obtained by Region-MA and Euclidean-MA (for $\epsilon = 10^{-5}$)

R+	R-	
Region-MA	Euclidean-MA	p-value
189	21	0.0008

563 5.2. Using the archive to reduce the search space

564 The archive is used to store solutions considered as optima to allow the
 565 algorithm to remove them from the population without losing them. In our
 566 algorithm, it is used also to mark some regions as areas excluded for the search.
 567 In this section, we are interested in assessing how using the regions represented
 568 in the archive as excluded areas for the exploration of the EA improves the
 569 exploration of the search space and thus the discovery of more optima.

570 In order to perform this comparison, we ran two versions of the algorithm.
 571 The first one is as presented in Section 3. The second one is the same algorithm
 572 without verifying that each solution created by the EA is present or not in
 573 the archive (steps 4-7 in Algorithm 2 are ignored). We thus compare here the
 574 proposed algorithm which uses an excluding archive (RMAwA) against one with
 575 a simple archive called RMA with Simple Archive (RMAwSA).

576 As in the previous experiment, we will only focus on the highest level of
 577 accuracy ($\epsilon = 10^{-5}$). Indeed, the specific use of the archive mainly affects
 578 the algorithm's ability to explore the search space and not the precision of the
 579 solutions obtained.

Table 6: PRs of the RMA using an excluding archive (RMAwA) and a simple archive (RMAwSA) for $\epsilon = 10^{-5}$ and computational time difference between the two versions.

Problem	F_1	F_2	F_3	F_4	F_5
RMAwA	1.000	1.000	1.000	1.000	1.000
RMAwSA	1.000	0.312	1.000	1.000	1.000
Time difference (%)	22.6	23.3	7.7	15.5	3.1
Problem	F_6	F_7	F_8	F_9	F_{10}
RMAwA	0.000	0.917	0.824	0.513	1.000
RMAwSA	0.000	0.658	0.908	0.343	0.983
Time difference (%)	46.3	34.8	50.8	43.4	4.1
Problem	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
RMAwA	1.000	1.000	0.997	0.813	0.703
RMAwSA	0.667	0.930	0.667	0.667	0.648
Time difference (%)	5.8	1.5	2.2	21.8	15.3
Problem	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}
RMAwA	0.670	0.660	0.233	0.128	0.125
RMAwSA	0.667	0.323	0.183	0.125	0.125
Time difference (%)	5.0	14.1	2.4	0.7	1.2

580 In Table 6, we show the *PRs* obtained by both versions of the algorithm and
 581 the time difference. Thanks to the excluding property of the archive, the per-
 582 formances of the algorithm are significantly improved (see Table 7 for Wilcoxon
 583 comparison). We also display in this table the CPU time increase caused by
 584 the use of the archive in the search. As we could have expected, this prop-
 585 erty implies more computational effort. However, the percentage increase in the

586 computational time is reduced with the complexity and the dimensionality of
 587 the problem. This can be easily explained by the fact that in higher dimensions,
 588 the computational time of the evaluation increases while the time cost of the
 589 archive remains steady regardless the dimensionality. Also, considering the sum
 590 of the computational time for the whole benchmark, the runtime of RMAwA
 591 is 8.2% higher than RMAwSA's (it cannot be calculated from table 6 because
 592 some functions take longer than others).

Table 7: Wilcoxon comparison of the *PR* of the RMA with and without archive (for $\epsilon = 10^{-5}$)

R+	R-	p-value
RMAwA	RMAwSA	
186.5	23.5	0.00132

593 5.3. Diversity and Exploration

594 In this section we analyse how RMAwA explores the search domain. First,
 595 we are going to study how the population diversity evolves along the search.
 596 Then, we visually analyse the exploration of the algorithm by plotting for several
 597 functions the solutions generated during the exploration phase.

598 5.3.1. Population diversity: Influence of the Number of Divisions

599 In this section we analyse the evolution of the population diversity during
 600 the search, and the influence of ND over the diversity. To do so, additional runs
 601 have been carried out and a diversity measure has been applied to the solutions
 602 into the population. The diversity measure applied is the following:

$$603 \text{ Diversity}_{Pop} = \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^N \text{dist}(x_i, x_j)}{N \cdot (N-1)/2} \quad (8)$$

604 where Pop is the current population, N is the population size, $dist$ is the
 605 Euclidean distance, and x_i, x_j are solutions in the population.

606 To study the influence of the current ND over the diversity, we are going
 607 to visualise and compare the diversity of the proposal (using the adaptive ND
 608 mechanism described in 3.2), with using a fixed ND.

609 Figure 3 shows the evolution of the diversity for functions F_7 , F_{16} and F_{18} .
 610 These functions have been selected for being representative of the different be-
 611 haviours detected in this benchmark. In axis x there is the number of evalua-
 612 tions, and in axis y the diversity measure. The vertical lines mark the updates
 613 of number of divisions (it only has influence over the adaptive ND version),
 614 dividing the axis x in five stages of the algorithm (each stage using a different
 ND). In the following, we are going to describe the main tendencies:

- 615 1. In functions with a small dimension, like F_7 (where $D=2$), we can observe
 616 two phases. In the initial stages of the search ($ND=2$, $ND=4$), because

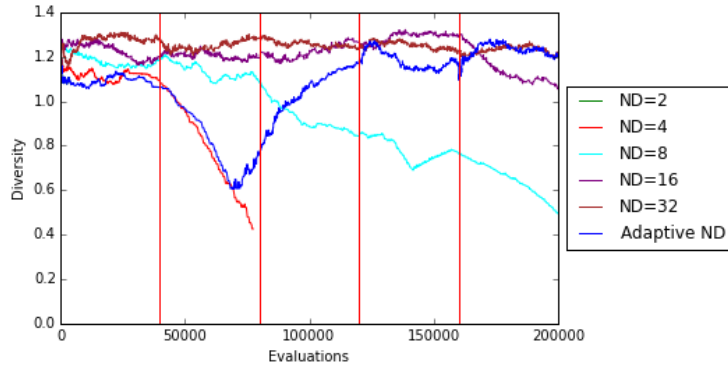
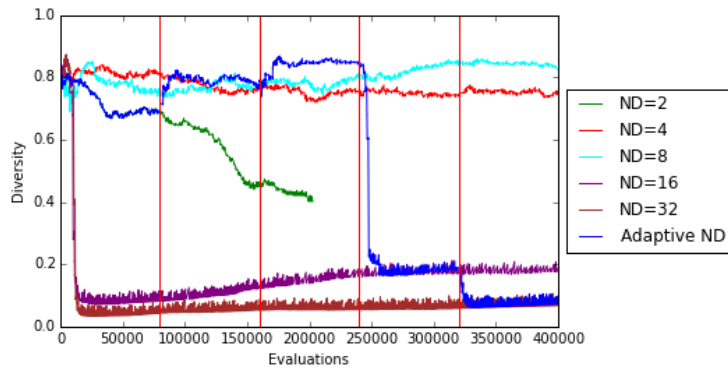
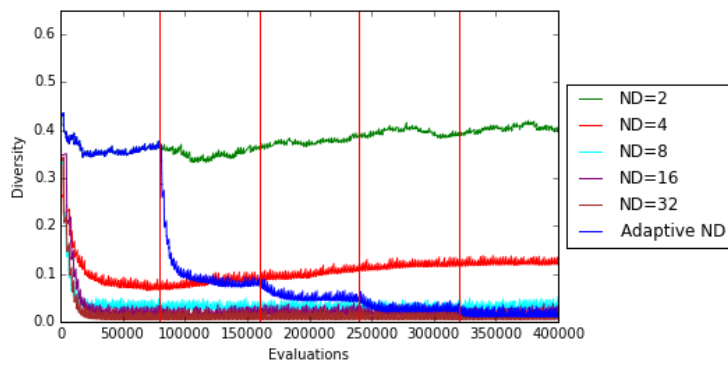
(a) F_7 (b) F_{16} (c) F_{18}

Figure 3: Diversity of the RMAWA population using adaptive number of divisions and using different fixed number of divisions during one run

617 there are few regions, when a region is avoided the search space is reduced
 618 very quickly to a small portion of the whole space, thus the diversity
 619 decreases very quickly. Indeed, for these ND values the fixed ND version
 620 prematurely stops because all possible regions have a local optima. The
 621 subsequent updates in ND increase the number of regions, releasing space
 622 for the EA to explore and thus increasing the diversity. As compared with
 623 fixed ND, the diversity of adaptive ND is very similar in the first two and
 624 final stages, with a greater diversity in the stages inbetween.

- 625 2. In functions with medium dimensionality, like F_{16} ($D=5$), the same phe-
 626 nomena is observed. However, after reaching a certain number of divisions
 627 per dimensions (third update) the diversity decreases, because the algo-
 628 rithm allows solutions more closer between them, reducing the diversity
 629 to enforce the exploitation of found solutions. Comparing adaptive ND
 630 with fixed ND, we can observe that diversity adaptive ND is actually very
 631 similar to ND in each stage.
- 632 3. In functions with higher dimensionality, like F_{18} ($D=10$), we can see that
 633 the diversity constantly decreases at each increase of the number of divi-
 634 sions. In these functions, it seems that the niching model does not provide
 635 a good balance in the population diversity during the search. Comparing
 636 adaptive ND with fixed ND, we can observe that adaptive ND obtains
 637 very close results to obtained by the fixed ND in each stage.

638 The previous section has shown the diversity differences comparing several
 639 fixed ND and the proposed dynamic ND. However, diversity itself is not our
 640 goal, thus we are going to compare the obtained PRs for each case. Table 8
 641 show the results, highlighting the results for those functions whose diversity has
 642 been analysed. We can observe that:

- 643 • In functions with a small dimension, like F_7 , in which a higher ND implies
 644 a better diversity, the number of optima increases also with the ND. Better
 645 results are obtained with dynamic ND.
- 646 • In functions with medium dimensionality, like F_{16} , in which for certain
 647 ND values the diversity is reduced very quickly, the PR decreases when
 648 ND increases. Dynamic ND, on the contrary, obtains the best PR value.
- 649 • Results obtained in functions with higher dimensionality, like F_{18} , proves
 650 that there is noy a good balance in the diversity, and that it has bad
 651 consequences for the obtained PR. In this case, dynamic ND obtains worse
 652 results than using $ND=2$ but better than the other values.

653 In summary, Figure 3 shows that the number of regions and problem di-
 654 mensionality have strong influence over the diversity in the population and the
 655 number of found optima, and that diversity using an adaptive ND is very close
 656 to that obtained with a fixed ND in each stage, obtaining the most robust
 657 behaviour when finding the optima.

Table 8: PRs of the RMAWA using different fixed numbers of divisions (ND) and with dynamic ND.

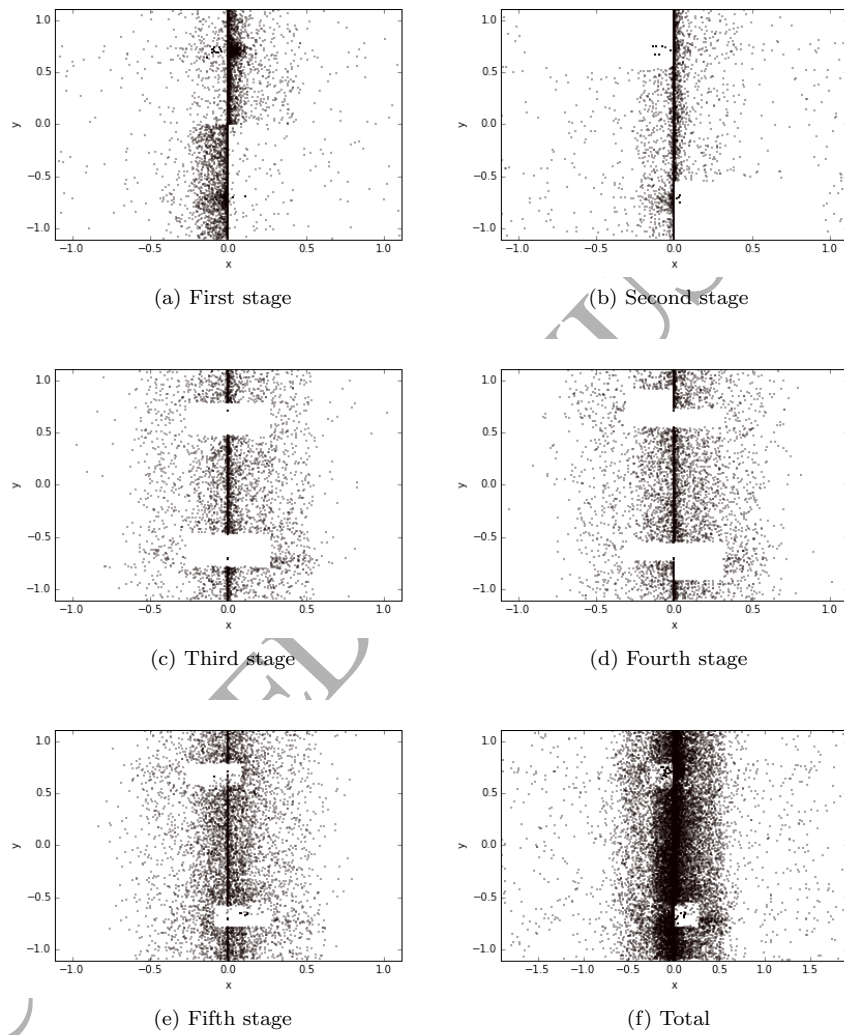
Function	ND=2	ND=4	ND=7	ND=12	ND=21	Dynamic ND
F1	0.900	1.000	1.000	1.000	1.000	1.000
F2	1.000	1.000	1.000	1.000	1.000	1.000
F3	1.000	1.000	1.000	1.000	1.000	1.000
F4	0.750	1.000	1.000	1.000	1.000	1.000
F5	1.000	1.000	1.000	1.000	1.000	1.000
F6	0.000	0.000	0.000	0.000	0.000	0.000
F7	0.084	0.429	0.612	0.790	0.829	0.917
F8	0.023	0.290	0.610	0.458	0.853	0.824
F9	0.035	0.172	0.433	0.660	0.618	0.513
F10	0.923	1.000	1.000	1.000	1.000	1.000
F11	0.733	1.000	1.000	1.000	1.000	1.000
F12	0.470	0.840	0.875	0.955	1.000	1.000
F13	0.680	0.993	1.000	1.000	1.000	0.997
F14	0.760	0.813	0.940	0.727	0.647	0.813
F15	0.665	0.725	0.675	0.640	0.275	0.703
F16	0.667	0.667	0.533	0.300	0.273	0.670
F17	0.660	0.680	0.250	0.185	0.165	0.660
F18	0.473	0.167	0.167	0.167	0.167	0.233
F19	0.160	0.125	0.125	0.125	0.125	0.128
F20	0.125	0.125	0.125	0.125	0.125	0.125
Mean	0.555	0.651	0.667	0.657	0.654	0.729

658 *5.3.2. Exploration of the domain search*

659 In this section, we study the exploration over the search space that RMAwA
660 carries out. First, we observe the solutions generated for each stage of the algo-
661 rithm to visualise the influence of the number of divisions over the exploration.
662 Then, we analyse if the exploration of the domain search is adapted to the
663 landscape of the function to optimise.

664 Figures 4, 5, and 6 show the generated and evaluated solutions by RMAwA
665 for the 2-D functions: f_5 , f_6 , and f_7 . Solutions generated by the LS have been
666 excluded, because they were too similar to previous solutions to be useful for
667 the analysis. In order to explore the influence of the current ND value over
668 the degree of exploration, in each figure the generated solutions for each stage
669 are shown differently (when the same ND value is applied). From these figures,
670 several conclusions can be extracted:

- 671 • In the initial stage the distribution of solutions is around the complete
672 domain search. There are two reasons for this: First, the initial population
673 has been randomly generated. Also, while there are no detected local
674 optima in one region, the new solutions are evaluated to check if they
675 have better fitness than the existing ones.
- 676 • In the following stages, several solutions have been detected as local op-
677 tima, so no more solutions are generated in the same regions. Thus, the
678 exploration shows several empty spaces around the detected optima.
- 679 • While the ND value decreases, these empty spaces are reduced, generating
680 solutions closer to current local optimum.
- 681 • In subfigures (f) with all the generated solutions, regions can be visualised
682 but not very clearly because they contain solutions generated in the first
683 stage, previous to the detection of local optima.

Figure 4: Generated solutions in function F_5 for each stage

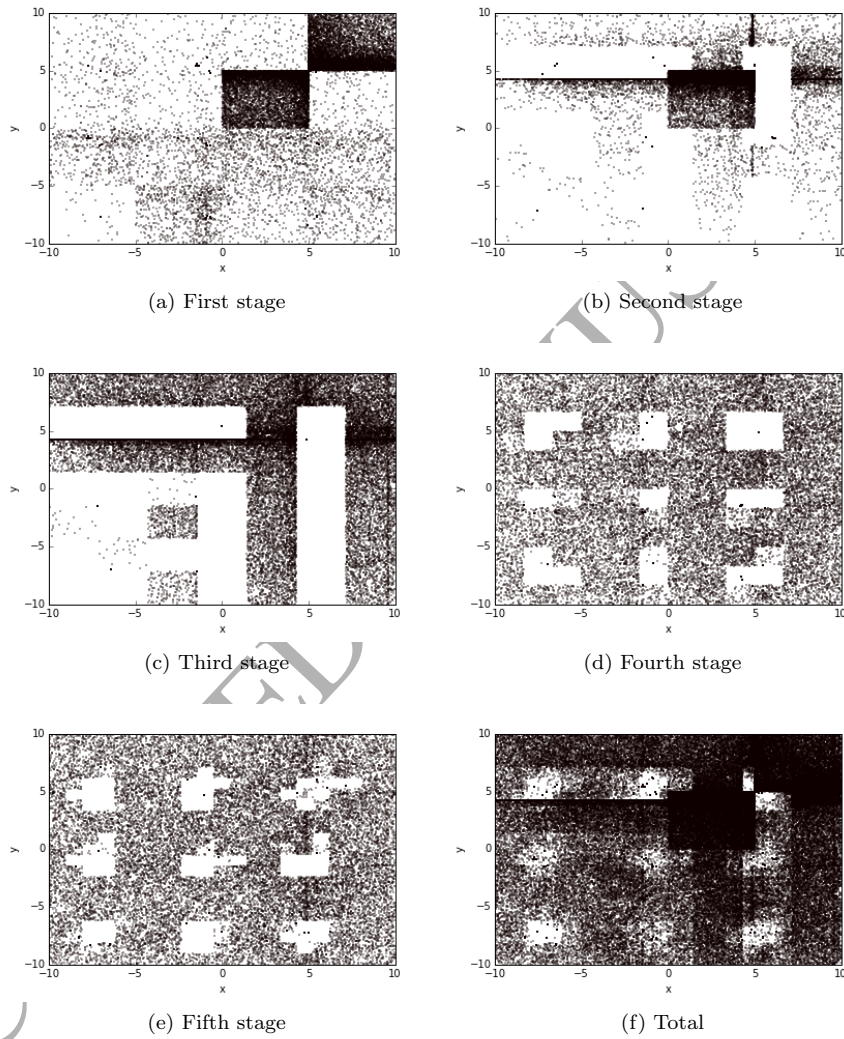
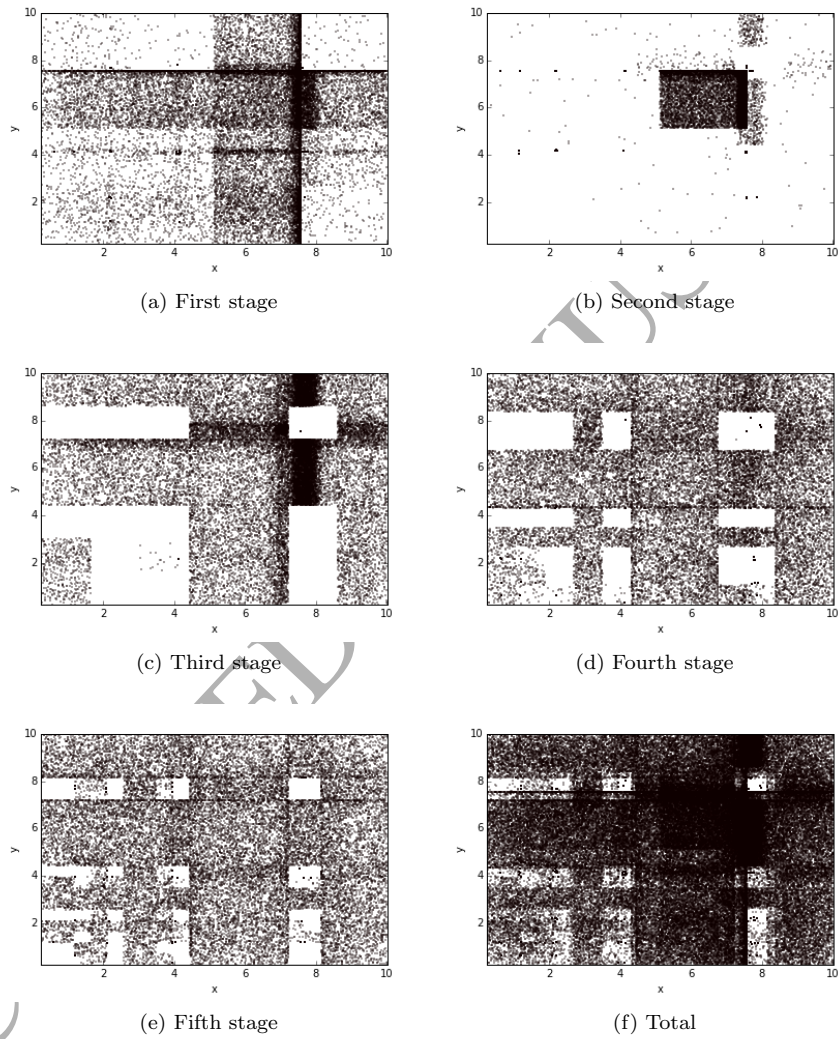


Figure 5: Generated solutions in function F_6 for each stage

Figure 6: Generated solutions in function F_7 for each stage

684 In order to show the exploration done for the algorithm, we plot for functions
 685 F_5 , F_6 and F_7 in Figures 7, 8, 9 respectively, the total solutions generated and
 686 evaluated by the algorithm (no using the LS method). To help the analysis, the
 687 contour of the studied functions are also plotted.

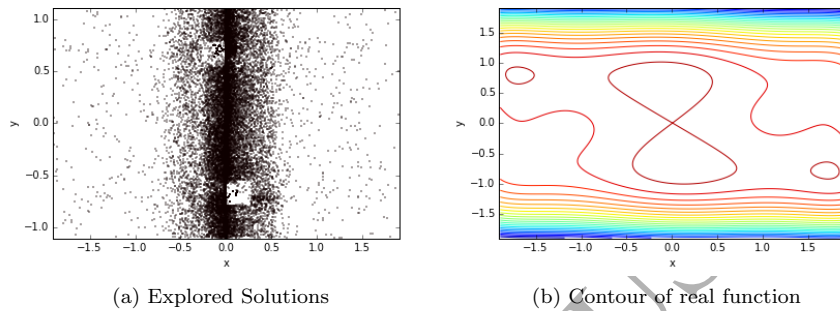


Figure 7: Function F_5

688 In Figure 7, we can see that all the domain search is explored, even when
 689 the best values are concentrated in one particular area. Also, the area close to
 690 each optimum has a reduced number of solutions, because the algorithm has
 691 identified them as optima and the region niching avoids solutions in the same
 692 region.

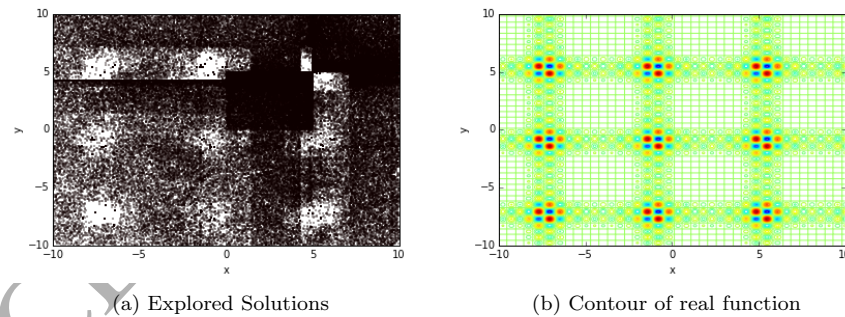
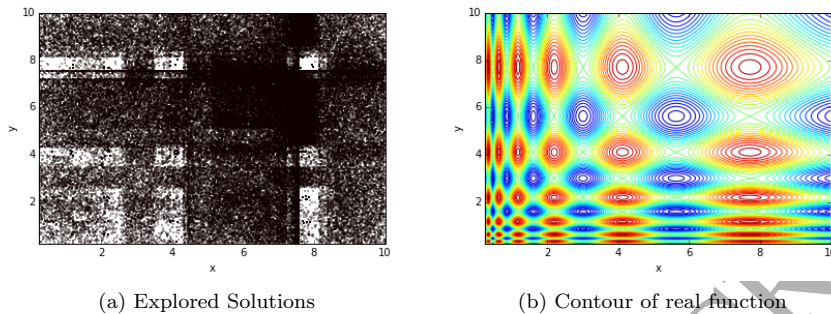


Figure 8: Function F_6

693 In Figures 8 and 9 we can observe the same behaviour, showing less dense
 694 areas surrounding each optimum, thus concentrating the majority of the solu-
 695 tions in regions with no detected optima. It is remarkable that the area of the
 696 landscape with the optima have been correctly identified.

697 In summary, the pattern drawn by the solutions generated during the explora-
 698 tion phase matches the landscape of the objective function. Also, RMAwA
 699 behaves as expected: exploring around all the domain search and avoiding at the

Figure 9: Function F_7

700 same time solutions which are very close to detected optima, defining regions
 701 with decreasing size.

702 5.4. Time and memory cost of RMAwA

703 In this section, we study the time and memory cost of RMAwA. First, we
 704 assess the memory used by the archive. Then we study the computational cost
 705 implied by the exclusive property of the archive and the different components
 706 of the algorithm.

707 5.4.1. Memory cost

708 We present in this section the memory cost implied by the archive. As ex-
 709 plained in Section 3.5, the archive list stores two kinds of solutions, the starting
 710 and final points of LS applications. In order to evaluate the memory cost of
 711 the archive in both cases, we retrieved the number of solutions stored in the
 712 archive's list and the number of their corresponding regions represented in the
 713 index at the end of each run. From these data, we estimate the total memory
 714 size of the archive. The archive's list is a collection of real-value vectors and the
 715 index is a collection of integer vectors. In our implementation, real values are
 716 represented by "double", coded on eight bytes and integers are represented by
 717 "int" coded on four bytes, the space used by the archive is thus calculated by:

$$ArchiveSize = |S| \cdot D \cdot 8 + |R| \cdot D \cdot 4 \quad (9)$$

718 where $|S|$ is the number of solutions in the archive's list, $|R|$ is the number
 719 of regions in the index and D is the dimensionality of the problem. The final
 720 size is thus proportionate to the dimensionality. It is also dependant on the
 721 maximum number of evaluations allowed by the problem. Indeed, an increase
 722 in the number of evaluations increases the number of LS applications and thus
 723 the number of solutions stored in the archive. In Table 9, we present the average
 724 of 50 runs of these data along with the dimensionality and the maximum number
 725 of evaluation for each function of the CEC'2013 benchmark.

726 As expected, we can observe a strong increase of the physical size used
 727 by the archive for the most complex problems. However, the memory used
 728 remains reasonable for today's machines. In the most extreme problem, F_{20}
 729 where $D = 20$, the archive only uses 64.88 kB of memory. Even if it might appear
 730 irrelevant for such problems, the size of the archive can increase exponentially
 731 with the dimensionality and the number of evaluation. When tackling large
 732 scale problems, one may consider limiting the size of the archive.

Table 9: Average number of elements in the archive's list ($|S|$), the index ($|R|$) and total memory used by the archive (in kB) at the end of each run

Problem	D	Max_{FEs}	$ S $	$ R $	ArchiveSize
F_1	1	$5.00 \cdot 10^4$	135.92	4.58	1.08
F_2	1	$5.00 \cdot 10^4$	130.24	9.96	1.06
F_3	1	$5.00 \cdot 10^4$	129.32	10.52	1.05
F_4	2	$5.00 \cdot 10^4$	106	22.76	1.83
F_5	2	$5.00 \cdot 10^4$	112.76	14.5	1.88
F_6	2	$2.00 \cdot 10^5$	425.52	112.64	7.53
F_7	2	$2.00 \cdot 10^5$	448.28	100.18	7.79
F_8	3	$4.00 \cdot 10^5$	681.84	398.62	20.65
F_9	3	$4.00 \cdot 10^5$	811.64	389.08	23.58
F_{10}	2	$2.00 \cdot 10^5$	431.28	100.68	7.53
F_{11}	2	$2.00 \cdot 10^5$	372.72	106.42	6.66
F_{12}	2	$2.00 \cdot 10^5$	326.04	104.42	5.91
F_{13}	2	$2.00 \cdot 10^5$	349.52	121.84	6.41
F_{14}	3	$4.00 \cdot 10^5$	583	283.48	16.99
F_{15}	3	$4.00 \cdot 10^5$	581.6	278.68	16.90
F_{16}	5	$4.00 \cdot 10^5$	524	259.42	25.54
F_{17}	5	$4.00 \cdot 10^5$	516.64	270.26	25.46
F_{18}	10	$4.00 \cdot 10^5$	446.84	187.36	42.23
F_{19}	10	$4.00 \cdot 10^5$	338.52	168.28	33.02
F_{20}	20	$4.00 \cdot 10^5$	343.8	142.92	64.88

733 5.4.2. Computational time of the different components of RMAwA

734 In this section, we analyse the amount of time taken by the different com-
 735 ponents of RMAwA over a whole run, namely:

- 736 • LS operations: the operations performed by CMA-ES during its search
 737 process.
- 738 • EA operations: the operations performed by the SSGA to evolve the pop-
 739 ulation.
- 740 • Niching: the time it takes for a new solution to go through the niching
 741 process (retrieval and comparison of the solutions present in the same

742 region in the population).

- 743 • Archive: the time implied by the excluding property of the archive (as-
- 744 assessing the presence of the solution's region in the archive's index).

745 First, to assess the computational time of each component, we use function
 746 f_{12} . This function presents the advantage of being implemented in 4 dimensions,
 747 $D = \{3, 5, 10, 20\}$, allowing us to evaluate the scalability of the proposal. For
 748 those four problems, we calculate the CPU time used by each component to
 749 assess their scalability. The search effort is unequally divided between the LS
 750 and the EA (the number of evaluation at each EA application is fixed while the
 751 number of evaluation for each LS application is not limited). Thus, to perform
 752 a fair comparison, we only select the average time per evaluation. We plot the
 753 results in Figure 10.

754 As far as we can see, the complexity of the niching strategy and the use of
 755 the archive are barely affected by an increase of the dimensionality. In the same
 756 way, the operations of the SSGA algorithms show interesting scalable properties.
 757 The main weakness lies in the use of CMA-ES as LS method. Although it offers
 758 a low complexity in the lowest dimensions, with more than ten variables, CMA-
 759 ES shows poor scalability in terms of complexity.

760 In order to counterbalance the importance of this drawback, we show in ta-
 761 ble 10 the CPU time of each of the components along with the evaluation time.
 762 Here, we remind the reader of the notation used in this paper, we grouped the
 763 problems F_j by function f_i in order to make for easier reading and see the rela-
 764 tions between the different dimensions of each function. From this table, when
 765 increasing the dimensionality, even if the proportion of the LS (*i.e.* CMA-ES)
 766 operations increases, the total CPU time is particularly affected by the compu-
 767 tational time of the evaluation which is independent of the algorithm. However,
 768 as the complexity of CMA-ES increases exponentially with the dimension, larger
 769 scale problems may require the use of another LS method.

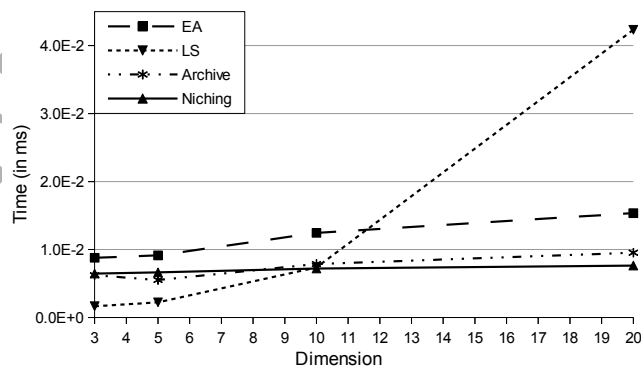


Figure 10: CPU time (in ms) of each component per evaluations for problem f_{12} for different dimensions

Table 10: CPU time (in seconds) details of RMAWA for each problem $F_j = \{f_i, D\}$ with the percentage in the whole optimisation process

Problem	$F_1 = \{f_1, 1\}$	$F_2 = \{f_2, 1\}$	$F_3 = \{f_3, 1\}$	$F_4 = \{f_4, 2\}$
Archive	0.150 (18.41%)	0.355 (40.84%)	0.148 (18.99%)	0.099 (14.96%)
Niching	0.236 (28.91%)	0.221 (25.39%)	0.254 (32.43%)	0.180 (27.17%)
EA	0.260 (31.86%)	0.257 (29.59%)	0.282 (36.08%)	0.207 (31.18%)
LS	0.164 (20.08%)	0.027 (3.07%)	0.064 (8.23%)	0.172 (25.93%)
Evaluations	0.006 (0.73%)	0.010 (1.11%)	0.033 (4.27%)	0.005 (0.76%)
Total	0.816	0.869	0.782	0.664
Problem	$F_5 = \{f_5, 2\}$	$F_6 = \{f_6, 2\}$	$F_8 = \{f_6, 3\}$	$F_7 = \{f_7, 2\}$
Archive	0.127 (9.96%)	1.085 (35.82%)	1.512 (29.74%)	1.354 (40.56%)
Niching	0.204 (16.01%)	0.759 (25.05%)	1.188 (23.37%)	0.787 (23.58%)
EA	0.233 (18.27%)	0.884 (29.18%)	1.513 (29.75%)	0.942 (28.23%)
LS	0.696 (54.67%)	0.159 (5.24%)	0.494 (9.72%)	0.173 (5.18%)
Evaluations	0.014 (1.10%)	0.143 (4.70%)	0.377 (7.41%)	0.081 (2.44%)
Total	1.273	3.029	5.085	3.338
Problem	$F_9 = \{f_7, 3\}$	$F_{10} = \{f_8, 2\}$	$F_{11} = \{f_9, 2\}$	$F_{12} = \{f_{10}, 2\}$
Archive	2.210 (35.27%)	0.740 (28.09%)	0.615 (6.15%)	0.521 (5.38%)
Niching	1.403 (22.38%)	0.759 (28.80%)	0.675 (6.74%)	0.586 (6.05%)
EA	1.793 (28.61%)	0.934 (35.41%)	0.903 (9.02%)	0.802 (8.28%)
LS	0.661 (10.55%)	0.153 (5.82%)	0.332 (3.32%)	0.207 (2.13%)
Evaluations	0.200 (3.19%)	0.050 (1.88%)	7.479 (74.77%)	7.571 (78.16%)
Total	6.267	2.636	10.003	9.687
Problem	$F_{13} = \{f_{11}, 2\}$	$F_{14} = \{f_{11}, 3\}$	$F_{16} = \{f_{11}, 5\}$	$F_{18} = \{f_{11}, 10\}$
Archive	0.604 (6.15%)	0.910 (3.88%)	0.853 (2.60%)	1.065 (1.81%)
Niching	0.630 (6.42%)	1.028 (4.39%)	0.951 (2.90%)	0.868 (1.48%)
EA	0.852 (8.69%)	1.441 (6.15%)	1.300 (3.96%)	1.267 (2.16%)
LS	0.210 (2.14%)	0.412 (1.76%)	0.671 (2.04%)	2.099 (3.58%)
Evaluations	7.515 (76.59%)	19.641 (83.82%)	29.054 (88.50%)	53.400 (90.97%)
Total	9.812	23.431	32.829	58.698
Problem	$F_{15} = \{f_{12}, 3\}$	$F_{17} = \{f_{12}, 5\}$	$F_{19} = \{f_{12}, 10\}$	$F_{20} = \{f_{12}, 20\}$
Archive	0.982 (4.15%)	0.773 (2.33%)	0.723 (1.22%)	0.891 (0.71%)
Niching	1.020 (4.31%)	0.932 (2.81%)	0.663 (1.12%)	0.714 (0.57%)
EA	1.387 (5.86%)	1.282 (3.87%)	1.144 (1.93%)	1.435 (1.15%)
LS	0.399 (1.69%)	0.580 (1.75%)	2.267 (3.81%)	12.974 (10.37%)
Evaluations	19.867 (83.98%)	29.560 (89.23%)	54.636 (91.93%)	109.063 (87.20%)
Total	23.655	33.127	59.434	125.077

770 *5.5. Comparison with existing algorithms*

771 In this section we compare the results obtained by our algorithm, RMAwA.
772 We selected a number of algorithms from the literature along with algorithms
773 presented for the CEC'2013 competition:

- 774 • PNA-NSGAI [2] proposed for the competition, this algorithm is an im-
775 provement of A-NSGAI [8]. These algorithms tackle the multimodal op-
776 timisation problem by turning them into bi-objective problems. The first
777 objective is the minimisation of the original function and the second one
778 is the maximisation of the diversity brought by the evaluated individual.
- 779 • dADE/nrand/1/bin [12] : a DE using a neighbourhood based mutation
780 strategy and a dynamically updated archive.
- 781 • DE/nrand/2 [13] : a DE using the neighbourhood based mutation strategy.
- 782 • NVMO [34]: a Variable Mesh optimisation algorithm with niching strat-
783 egy.
- 784 • CMA-ES [18]: A version of CMA-ES that implements a simple archive.
- 785 • NEA2 [43]: A version of CMA-ES that uses nearest-better clustering as
786 niching strategy.

787 These algorithms are the top six algorithms of the CEC'2013 competition.
788 All the results used here were provided by the authors and used during the
789 competition. The detailed results of each algorithm can be seen in the Appendix.
790 We first analyse the overall performance of each algorithm on the benchmark
791 and compare them with RMAwA. Then we study in detail their behaviour
792 according to the problem's characteristics.

793 *5.5.1. Accuracy level analysis*

794 We analyse here the general performance of these algorithms on the CEC'2013
795 benchmark for each accuracy level. To support this analysis, we show in Ta-
796 ble 11 the mean rankings of each algorithm according to the different accuracy
797 levels and in Table 12 the Wilcoxon comparison of RMAwA with the other
798 algorithms.

799 First, when comparing with other algorithms using CMA-ES, we can see that
800 RMAwA significantly outperforms the classical CMA-ES (with $\alpha = 0.1$). This
801 algorithm is not particularly designed for multimodal optimisation as it does
802 not implement any niching mechanism. When comparing with NEA2, RMAwA
803 offers similar performance.

804 Then, we can see that RMAwA is third best for the smallest accuracy level
805 ($\epsilon = 1E-1$) behind NVMO and dADE although no statistical difference can be
806 observed in Table 12.

807 Excluding NEA2, RMAwA obtains better results than the other algorithms,
808 and this superiority increases with the accuracy level, being specially remarkable

Table 11: Mean rankings obtained by different algorithms over all functions CEC'2013 benchmark for each accuracy level

Accuracy level	$1E-1$	$1E-2$	$1E-3$	$1E-4$	$1E-5$
PNA-NSGAI	4.53	5.18	5.28	5.45	5.43
DE/nrand/2	5.53	5.05	4.95	4.83	4.30
CMA-ES	4.58	4.00	4.08	3.93	3.90
NVMO	2.73	3.68	4.15	4.43	5.08
dADE	3.43	4.10	4.10	4.08	3.90
NEA2	3.70	2.75	2.55	2.68	2.70
RMAwA	3.53	3.25	2.90	2.63	2.70

809 for $\epsilon = \{1E-4, 1E-5\}$. Between NEA2 and RMAwA there is no statistical
810 difference detected.

811 This analysis highlights the difficulty of algorithms to properly balance the
812 exploration and the exploitation. Indeed, when algorithms use the original
813 CMA-ES, a very efficient method to obtain accurate solutions but also very
814 costly, they generally perform better for higher accuracy levels. On the other
815 hand, other algorithms (DE-based, NVMO, PNA-NSGAI) have better explo-
816 ration efficiency but fail to identify accurate solutions.

817 5.5.2. Problem specific performance analysis

818 Let us now consider every problem individually. As it is the most challenging
819 for this benchmark, we will consider here only the highest accuracy level ($\epsilon =$
820 $1E-5$). Table 13 lists the PRs obtained by each algorithm for this accuracy
821 level.

822 In this analysis we will focus on the problems offering the major differences
823 between the results obtained by the compared algorithms. Concerning problems
824 with highly multimodal fitness landscapes, F_7 to F_9 where the number of optima
825 ranges from 36 to 216, RMAwA ranks amongst the best algorithm. It obtains
826 the best results for problem F_7 and obtains the second best results of problem
827 F_8 and F_9 after respectively dADE and NEA2.

828 RMAwA also shows the best results in the problems with composition func-
829 tions (F_{10} to F_{17}). However the quality of the results decreases with the dimen-
830 sionality (F_{18} to F_{20}), being clearly worse than that obtained by NEA2.

831 The improvable behaviour of RMAwA when the dimensionality increases is
832 clear because for f_{11} and f_{12} results are very good with dimension 2, but not
833 good with a higher dimension, like 10 (F_{18} - F_{20}). In Section 5.3.1, it can be
834 observed that for these functions the results obtained are low for each possible
835 ND, thus the results are not due to the ND adaptation mechanism. Another
836 possible reason of the improvable results could be that the parameter values
837 have been automatically tuned considering all functions, in which the majority
838 has a very low dimension. Because of this, these parameter values could not be
839 the more adequate for lower dimension problems. In order to reject or confirm

Table 12: Wilcoxon comparison of the PRs of the RMAwA ($R+$) with other algorithms ($R-$) (for $\epsilon = \{1E-1, 1E-2, 1E-3, 1E-4, 1E-5\}$)

$\epsilon = 1E-1$			
RMAwA vs	$R+$	$R-$	p-value
PNA-NSGAI	128.5	65	2.27E-1
DE/nrand/2	180.5	29.5	3.40E-3
CMA-ES	168.5	41.5	1.62E-2
NVMO	62	132.5	1.96E-1
dADE	71.5	122	3.44E-1
NEA2	117.5	92.5	6.41E-1
$\epsilon = 1E-2$			
RMAwA vs	$R+$	$R-$	p-value
PNA-NSGAI	199.5	10.5	9.35E-5
DE/nrand/2	171.5	38.5	1.14E-2
CMA-ES	150.5	59.5	9.35E-2
NVMO	124.5	85.5	4.67E-1
dADE	134.5	75.5	2.71E-1
NEA2	82.5	127.5	4.01E-1
$\epsilon = 1E-3$			
RMAwA vs	$R+$	$R-$	p-value
PNA-NSGAI	185	7.5	8.39E-5
DE/nrand/2	171.5	38.5	1.14E-2
CMA-ES	139	53.5	9.98E-2
NVMO	166.5	43.5	2.04E-2
dADE	147.5	62.5	1.19E-1
NEA2	91.5	118.5	6.14E-1
$\epsilon = 1E-4$			
RMAwA vs	$R+$	$R-$	p-value
PNA-NSGAI	185	7.5	8.39E-5
DE/nrand/2	171.5	38.5	1.14E-2
CMA-ES	139	53.5	9.98E-2
NVMO	185.5	24.5	1.56E-3
dADE	165.5	44.5	2.27E-2
NEA2	95	97.5	1.00E+0
$\epsilon = 1E-5$			
RMAwA vs	$R+$	$R-$	p-value
PNA-NSGAI	199.5	10.5	9.35E-5
DE/nrand/2	151.5	42	3.23E-2
CMA-ES	138	54.5	1.09E-1
NVMO	189.5	20.5	7.79E-4
dADE	151.5	42	3.23E-2
NEA2	96.5	96	9.68E-1

Table 13: PRs obtained by each algorithm for $\epsilon = 1E-5$ on the CEC'2013 benchmark. Values in the parenthesis represent the standard competition ranking of each algorithm for each problem

Problem	PNA-NSGAIH	DE/mrand/2	CMMA-ES	NVMO	dADE	NEA2	RMAwA
F_1	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)
F_2	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)
F_3	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)
F_4	0.805 (7)	1.000 (1)	0.990 (5)	1.000 (1)	1.000 (1)	0.990 (5)	1.000 (1)
F_5	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)	1.000 (1)
F_6	0.000 (1)	0.000 (1)	0.000 (1)	0.000 (1)	0.000 (1)	0.000 (1)	0.000 (1)
F_7	0.683 (5)	0.275 (7)	0.516 (6)	0.804 (3)	0.714 (4)	0.911 (2)	0.917 (1)
F_8	0.252 (4)	0.363 (3)	0.115 (6)	0.027 (7)	0.947 (1)	0.239 (5)	0.824 (2)
F_9	0.276 (4)	0.065 (7)	0.272 (5)	0.194 (6)	0.349 (3)	0.579 (1)	0.513 (2)
F_{10}	1.000 (1)	1.000 (1)	0.978 (6)	0.967 (7)	1.000 (1)	0.980 (5)	1.000 (1)
F_{11}	0.663 (7)	0.667 (5)	0.953 (3)	0.667 (4)	0.667 (5)	0.960 (2)	1.000 (1)
F_{12}	0.573 (7)	0.618 (5)	0.760 (3)	0.593 (6)	0.735 (4)	0.833 (2)	1.000 (1)
F_{13}	0.623 (7)	0.667 (4)	0.947 (2)	0.663 (6)	0.667 (4)	0.947 (2)	0.997 (1)
F_{14}	0.610 (7)	0.667 (4)	0.743 (3)	0.627 (6)	0.667 (4)	0.800 (2)	0.813 (1)
F_{15}	0.443 (5)	0.400 (6)	0.653 (3)	0.378 (7)	0.620 (4)	0.713 (1)	0.703 (2)
F_{16}	0.323 (7)	0.667 (3)	0.663 (5)	0.653 (6)	0.667 (3)	0.673 (1)	0.670 (2)
F_{17}	0.245 (7)	0.280 (6)	0.583 (3)	0.325 (5)	0.358 (4)	0.695 (1)	0.660 (2)
F_{18}	0.093 (7)	0.507 (3)	0.340 (4)	0.327 (5)	0.603 (2)	0.663 (1)	0.233 (6)
F_{19}	0.010 (6)	0.180 (3)	0.597 (2)	0.093 (5)	0.000 (7)	0.667 (1)	0.128 (4)
F_{20}	0.000 (5)	0.230 (3)	0.425 (1)	0.000 (5)	0.000 (5)	0.350 (2)	0.125 (4)

that hypothesis, we have carried out another automatic tuning considering only functions $F_{18} - F_{20}$, but the results obtained were very similar. Thus, the improvable behaviour or RMAwA in higher dimensionality problems is kept as a open issue to be solved in the future.

However, this previous behaviour is not unsurprising, because it has happened to many others, as can be observed in Table 13. As was formulated by the No Free Lunch Theorem, designing an algorithm for an heterogeneous test bed of problems is very challenging. It is common for algorithms to perform well in problems with certain characteristics and poorly on others.

In summary, the algorithm proposed of this paper, RMAwA, offers an overall performance significantly superior to the other algorithms by obtaining competitive if not better results in most problems (except in higher dimension problems) proposed in the CEC'2013 benchmark. Only NEA2, the winner of the CEC'2013 competition obtains equivalent results.

6. Conclusions

In this paper, we present a novel model based on region-based MA to tackle multimodal optimisation problems. It uses a clearing strategy where niches are defined as regions. It implements an archive of solutions and indexed regions considered as explored and thus excluded from further exploration.

In order to asses the efficiency of the model against existing ones, we have tested it on a MA which alternatively applies an EA (SSGA) to explore the search space and an LS (CMA-ES) to the best one until it does not improve for a certain number of evaluations.

Various studies have been performed to study the performances of this model. First, we have demonstrated that the use of region-based niches was more efficient than that of the classical euclidean niches. We have shown that excluding regions explored by the LS allows the algorithm to reduce the search space leading to a more efficient exploration. Also, we have analysed the population diversity during the run and the degree of exploration in several functions. Finally, complexity testing show the good scalability of the proposal.

We compared the resulting algorithm using the benchmark issued for the special session and competition on niching methods for multimodal function optimisation of the IEEE Congress on Evolutionary Computation in 2013. We noted that our algorithm was fairly independent to the different accuracy levels tested in this benchmark compared to the other algorithms obtaining significantly better results than most algorithms and similar performance to NEA2.

This work opens the way of various potential future studies:

- First, the behaviour of RMAwA with higher dimension problems should be studied more in detailed and improved it. Also, the memory cost of the archive may become consequent when tackling higher dimension problems and it could be interesting to study techniques which reduce or limit the size of the archive or remove similar solutions representing the same region.

- 882 • As is often the case when the parameter defining the size of a niche, the
883 number of divisions per dimensions is highly sensitive. Although the idea
884 of a constant increasing during the search might offer interesting results,
885 it may not be optimal in some cases. A more adapted strategy could be
886 identified and researched. An other option would be to implement multi-
887 population where, as it is done in Hierarchical Genetic Strategy [53], each
888 population uses different numbers of divisions.
- 889 • This model has proved to obtain interesting results when applied with
890 both SSGA and CMAES as its EA and LS. Further experiments using
891 different components or in a different memetic framework could lead to
892 new efficient algorithms. If tested in higher dimensions, the necessity of
893 changing CMAES to a more scalable LS method would be compulsory.

894 Acknowledgements

895 This Work was supported by the Spanish Ministry of Education and Science
896 under Grants TIN2012-37930-C02-01, TIN2014-57251-P, TIN2013-47210-P and
897 Research Regional Projects P10-TIC-6858, P12-TIC-2958.

898 References

- 899 [1] C. F. Agostinho, C. Fernandes, A. Rosa, A Study on Non-random Mating
900 and Varying Population Size in Genetic Algorithms Using a Royal Road
901 Function, in: Proceedings of the 2001 IEEE Congress on Evolutionary
902 Computation, IEEE Press, 2001, pp. 60–66.
- 903 [2] S. Bandaru, K. Deb, A parameterless-niching-assisted bi-objective
904 approach to multimodal optimization, in: Evolutionary Computation (CEC),
905 2013 IEEE Congress on, 2013, pp. 95–102.
- 906 [3] S. Bird, X. Li, Adaptively choosing niching parameters in a PSO, in: Pro-
907 ceedings of the 8th annual conference on Genetic and evolutionary compu-
908 tation, GECCO '06, ACM, New York, NY, USA, 2006, pp. 3–10.
- 909 [4] S. Biswas, S. Kundu, S. Das, An improved parent-centric mutation with
910 normalized neighborhoods for inducing niching behavior in differential evo-
911 lution, Cybernetics, IEEE Transactions on 44 (10) (2014) 1726–1737.
- 912 [5] X. Chen, Y.-S. Ong, M.-H. Lim, K. C. Tan, A multi-facet survey on
913 memetic computation, Evolutionary Computation, IEEE Transactions on
914 15 (5) (2011) 591–607.
- 915 [6] S. Das, S. Maity, B.-Y. Qu, P. Suganthan, Real-parameter evolutionary
916 multimodal optimization a survey of the state-of-the-art, Swarm and Evo-
917 lutionary Computation 1 (2) (2011) 71 – 88.

- 918 [7] K. A. De Jong, An analysis of the behavior of a class of genetic adaptive
919 systems., Ph.D. thesis, Ann Arbor, MI, USA (1975).
- 920 [8] K. Deb, A. Saha, Multimodal optimization using a bi-objective evolution-
921 ary algorithm, *Evolutionary Computation* 20 (1) (2012) 27–62.
- 922 [9] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use
923 of nonparametric statistical tests as a methodology for comparing evolu-
924 tionary and swarm intelligence algorithms, *Swarm and Evolutionary Com-
925 putation* 1 (1) (2011) 3–18.
- 926 [10] G. Dick, P. Whigham, Weighted local sharing and local clearing for multi-
927 modal optimisation, *Soft Computing* 15 (9) (2011) 1707–1721.
- 928 [11] M. Ellabaan, Y.-S. Ong, Valley-adaptive clearing scheme for multimodal
929 optimization evolutionary search, in: *Intelligent Systems Design and Ap-
930 plications, 2009. ISDA '09. Ninth International Conference on, 2009*, pp.
931 1–6.
- 932 [12] M. Eptropakis, X. Li, E. Burke, A dynamic archive niching differential
933 evolution algorithm for multimodal optimization, in: *Evolutionary Com-
934 putation (CEC), 2013 IEEE Congress on, 2013*, pp. 79–86.
- 935 [13] M. Eptropakis, V. Plagianakos, M. Vrahatis, Finding multiple global op-
936 tima exploiting differential evolution's niching capability, in: *Differential
937 Evolution (SDE), 2011 IEEE Symposium on, 2011*, pp. 1–8.
- 938 [14] L. J. Eshelman, J. D. Schaffer, Real-coded Genetic Algorithms in Ge-
939 netic Algorithms by Preventing Incest, *Foundation of Genetic Algorithms*
940 2 (1993) 187–202.
- 941 [15] W. Gao, G. G. Yen, S. Liu, A cluster-based differential evolution with self-
942 adaptive strategy for multimodal optimization, *Cybernetics, IEEE Trans-
943 actions on* 44 (8) (2014) 1314–1327.
- 944 [16] D. E. Goldberg, J. Richardson, Genetic algorithms with sharing for multi-
945 modal function optimization, in: *Proceedings of the Second International
946 Conference on Genetic Algorithms on Genetic algorithms and their appli-
947 cation*, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1987, pp. 41–49.
- 948 [17] N. Hansen, S. D. Müller, P. Koumoutsakos, Reducing the Time Complexity
949 of the Derandomized Evolution Strategy with Covariance Matrix Adapta-
950 tion (CMA-ES), *Evolutionary Computation* 1 (11) (2003) 1–18.
- 951 [18] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in
952 evolution strategies, *Evolutionary Computation* 9 (2) (2001) 159–195.
- 953 [19] A. Kononova, D. Ingham, M. Pourkashanian, Simple scheduled memetic
954 algorithm for inverse problems in higher dimensions: Application to chemi-
955 cal kinetics, in: *Evolutionary Computation, 2008. CEC 2008. (IEEE World*

- 956 Congress on Computational Intelligence). IEEE Congress on, 2008, pp.
957 3905–3912.
- 958 [20] N. Krasnogor, J. Smith, A tutorial for competent memetic algorithms:
959 model, taxonomy, and design issues, IEEE Transactions on Evolutionary
960 Computation 9 (5) (2005) 474 – 488.
- 961 [21] B. Lacroix, D. Molina, F. Herrera, Region based memetic algorithm for
962 real-parameter optimisation, Information Sciences 262 (2014) 15 – 31.
- 963 [22] J.-P. Li, M. E. Balazs, G. T. Parks, P. J. Clarkson, A species conserving
964 genetic algorithm for multimodal function optimization, Evol. Comput.
965 10 (3) (2002) 207–234.
- 966 [23] X. Li, Adaptively choosing neighbourhood bests using species in a particle
967 swarm optimizer for multimodal function optimization, in: K. Deb (ed.),
968 Genetic and Evolutionary Computation GECCO 2004, vol. 3102 of Lecture
969 Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 105–116.
- 970 [24] X. Li, Niching without niching parameters: Particle swarm optimization
971 using a ring topology, Evolutionary Computation, IEEE Transactions on
972 14 (1) (2010) 150–169.
- 973 [25] X. Li, A. Engelbrecht, M. Epitropakis, Benchmark functions for cec 2013
974 special session and competition on niching methods for multimodal function
975 optimization, Tech. rep., Royal Melbourne Institute of Technology (2013).
- 976 [26] Li Xiaodong, A multimodal particle swarm optimizer based on fitness
977 euclidean-distance ratio, in: Proceedings of the 9th annual conference on
978 Genetic and evolutionary computation, GECCO '07, ACM, New York, NY,
979 USA, 2007, pp. 78–85.
- 980 [27] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, Automatic configuration of
981 state-of-the-art multi-objective optimizers using the TP+PLS framework,
982 in: N. Krasnogor, P. L. Lanzi (eds.), Proceedings of the Genetic and Evo-
983 lutionary Computation Conference, GECCO 2011, ACM Press, New York,
984 NY, 2011, pp. 2019–2026.
- 985 [28] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, A hybrid TP+PLS algo-
986 rithm for bi-objective flow-shop scheduling problems, Computers & Oper-
987 ations Research 38 (8) (2011) 1219–1236.
- 988 [29] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, M. Birattari, The irace
989 package, iterated race for automatic algorithm configuration, Tech. Rep.
990 TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium
991 (2011).
- 992 [30] S. W. Mahfoud, Niching methods for genetic algorithms, Ph.D. thesis,
993 Champaign, IL, USA (1995).

- 994 [31] O. J. Mengshoel, D. E. Goldberg, Probabilistic crowding: Deterministic
995 crowding with probabilistic replacement, in: Proceedings of the Genetic
996 and Evolutionary Computation Conference (GECCO-99), 1999, pp. 409–
997 416.
- 998 [32] D. Molina, M. Lozano, C. García-Martínez, F. Herrera, Memetic Algo-
999 rithms for Continuous Optimisation Based on Local Search Chains, *Evolutionary Computation* 18 (1) (2010) 27–63.
- 1000
- 1001 [33] D. Molina, M. Lozano, A. M. Sánchez, F. Herrera, Memetic algorithms
1002 based on local search chains for large scale continuous optimisation prob-
1003 lems: MA-SSW-Chains, *Soft Computing* 15 (11) (2010) 2201–2220.
- 1004 [34] D. Molina, A. Puris, R. Bello, F. Herrera, Variable mesh optimization for
1005 the 2013 CEC Special Session Niching Methods for Multimodal Optimiza-
1006 tion, in: *Evolutionary Computation (CEC), 2013 IEEE Congress on*, 2013,
1007 pp. 87–94.
- 1008 [35] P. Moscato, *On Evolution, Search, Optimization, Genetic Algorithms and*
1009 *Martial Arts: Toward Memetic Algorithms*, Tech. rep. (1989).
- 1010 [36] H. Mülenbein, D. Schlierkamp-Voosen, Predictive models for the breed-
1011 ing genetic algorithm in continuous parameter optimization, *Evolutionary*
1012 *Computation* 1 (1993) 25–49.
- 1013 [37] F. Neri, C. Cotta, Memetic algorithms and memetic computing optimiza-
1014 tion: A literature review, *Swarm and Evolutionary Computation* 2 (2012)
1015 1 – 14.
- 1016 [38] F. Neri, V. Tirronen, T. Karkkainen, T. Rossi, Fitness diversity based
1017 adaptation in multimeme algorithms: a comparative study, in: *Evolutionary*
1018 *Computation, 2007. CEC 2007. IEEE Congress on*, 2007, pp. 2374–2381.
- 1019 [39] Y.-S. Ong, A. Keane, Meta-lamarckian learning in memetic algorithms,
1020 *Evolutionary Computation, IEEE Transactions on* 8 (2) (2004) 99–110.
- 1021 [40] Y.-S. Ong, M.-H. Lim, N. Zhu, K.-W. Wong, Classification of adaptive
1022 memetic algorithms: a comparative study, *Systems, Man, and Cybernetics,*
1023 *Part B: Cybernetics, IEEE Transactions on* 36 (1) (2006) 141–152.
- 1024 [41] L. Pérez Cáceres, M. López-Ibáñez, T. St utzle, An analysis of parameters of
1025 IRACE, in: C. Blum, G. Ochoa (eds.), *Evolutionary Computation in Com-*
1026 *binatorial Optimisation*, vol. 8600 of *Lecture Notes in Computer Science*,
1027 Springer Berlin Heidelberg, 2014, pp. 37–48.
- 1028 [42] A. Petrowski, A clearing procedure as a niching method for genetic algo-
1029 rithms, in: *Evolutionary Computation, 1996., Proceedings of IEEE Inter-*
1030 *national Conference on*, 1996, pp. 798–803.

- 1031 [43] M. Preuss, Niching the CMA-ES via nearest-better clustering, in: Pro-
1032 ceedings of the 12th Annual Conference Companion on Genetic and Evolu-
1033 tionary Computation, GECCO '10, ACM, New York, NY, USA, 2010, pp.
1034 1711–1718.
- 1035 [44] L. Qing, W. Gang, Y. Zaiyue, W. Qiuping, Crowding clustering genetic
1036 algorithm for multimodal function optimization, *Applied Soft Computing*
1037 8 (1) (2008) 88 – 95.
- 1038 [45] B. Qu, P. Gouthanan, P. Suganthan, Dynamic grouping crowding differen-
1039 tial evolution with ensemble of parameters for multi-modal optimization,
1040 in: B. Panigrahi, S. Das, P. Suganthan, S. Dash (eds.), *Swarm, Evolu-
1041 tionary, and Memetic Computing*, vol. 6466 of *Lecture Notes in Computer
1042 Science*, Springer Berlin Heidelberg, 2010, pp. 19–28.
- 1043 [46] B. Qu, J. Liang, P. Suganthan, Niching particle swarm optimization with
1044 local search for multi-modal optimization, *Information Sciences* 197 (2012)
1045 131 – 143.
- 1046 [47] B. Qu, J. Liang, P. Suganthan, T. Chen, Ensemble of clearing differential
1047 evolution for multi-modal optimization, in: Y. Tan, Y. Shi, Z. Ji (eds.),
1048 *Advances in Swarm Intelligence*, vol. 7331 of *Lecture Notes in Computer
1049 Science*, Springer Berlin Heidelberg, 2012, pp. 350–357.
- 1050 [48] B. Y. Qu, P. Suganthan, J. Liang, Differential evolution with neighborhood
1051 mutation for multimodal optimization, *Evolutionary Computation, IEEE
1052 Transactions on* 16 (5) (2012) 601–614.
- 1053 [49] S. Bird and Li Xiaodong, Enhancing the robustness of a speciation-based
1054 PSO, in: *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on,
1055 2006*, pp. 843–850.
- 1056 [50] W. F. Sacco, M. D. Machado, C. M. Pereira, R. Schirru, The fuzzy clearing
1057 approach for a niching genetic algorithm applied to a nuclear reactor core
1058 design optimization problem, *Annals of Nuclear Energy* 31 (1) (2004) 55 –
1059 69.
- 1060 [51] B. Sareni, L. Krahenbuhl, Fitness sharing and niching methods revisited,
1061 *Evolutionary Computation, IEEE Transactions on* 2 (3) (1998) 97–106.
- 1062 [52] R. Schaefer, K. Adamska, H. Telega, Clustered genetic search in continuous
1063 landscape exploration, *Engineering Applications of Artificial Intelligence*
1064 17 (4) (2004) 407–416.
- 1065 [53] R. Schaefer, J. Kolodziej, Genetic search reinforced by the population hi-
1066 erarchy, in: K. A. D. Jong, R. Poli, J. E. Rowe (eds.), *FOGA, Morgan
1067 Kaufmann*, 2002, pp. 383–400.

- 1068 [54] J. Smith, Coevolving memetic algorithms: A review and progress report,
1069 Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions
1070 on 37 (1) (2007) 6–17.
- 1071 [55] C. Stoean, M. Preuss, R. Stoean, D. Dumitrescu, Ea-powered basin num-
1072 ber estimation by means of preservation and exploration, in: G. Rudolph,
1073 T. Jansen, S. Lucas, C. Poloni, N. Beume (eds.), Parallel Problem Solving
1074 from Nature PPSN X, vol. 5199 of Lecture Notes in Computer Science,
1075 Springer Berlin Heidelberg, 2008, pp. 569–578.
- 1076 [56] H. Telega, R. Schaefer, E. Cabib, A parallel genetic clustering for inverse
1077 problems, in: B. Kgstrm, J. Dongarra, E. Elmroth, J. Waniewski (eds.),
1078 Applied Parallel Computing Large Scale Scientific and Industrial Problems,
1079 vol. 1541 of Lecture Notes in Computer Science, Springer Berlin Heidelberg,
1080 1998, pp. 551–556.
- 1081 [57] R. Thomsen, Multimodal optimization using crowding-based differential
1082 evolution, in: Evolutionary Computation, 2004. CEC2004. Congress on,
1083 vol. 2, 2004, pp. 1382–1389 Vol.2.
- 1084 [58] M. Tomassini, Spatially Structured Evolutionary Algorithms, Springer,
1085 2005.
- 1086 [59] M. Črepinšek, S.-H. Liu, M. Mernik, Exploration and exploitation in evo-
1087 lutionary algorithms: A survey, ACM Comput. Surv. 45 (3) (2013) 35:1–
1088 35:33.
- 1089 [60] J. E. Vitela, O. Castanos, A real-coded niching memetic algorithm for con-
1090 tinuous multimodal function optimization, in: Evolutionary Computation,
1091 2008. CEC 2008. (IEEE World Congress on Computational Intelligence).
1092 IEEE Congress on, 2008, pp. 2170–2177.
- 1093 [61] J. E. Vitela, O. Castaos, A sequential niching memetic algorithm for con-
1094 tinuous multimodal function optimization, Applied Mathematics and Com-
1095 putation 218 (17) (2012) 8242 – 8259.
- 1096 [62] H. Wang, I. Moon, S. Yang, D. Wang, A memetic particle swarm optimiza-
1097 tion algorithm for multimodal optimization problems, Information Sciences
1098 197 (2012) 38 – 52.
- 1099 [63] D. Zaharie, Extensions of differential evolution algorithms for multimodal
1100 optimization, in: Proceedings of SYNASC, 2004, pp. 523–534.
- 1101 [64] Z. Zhai, X. Li, A dynamic archive based niching particle swarm optimizer
1102 using a small population size, in: Proceedings of the Thirty-Fourth Aus-
1103 tralasian Computer Science Conference - Volume 113, ACSC'11, Australian
1104 Computer Society, Inc., Darlinghurst, Australia, Australia, 2011, pp. 83–
1105 90.

1106 **Appendix: Detailed Peak Ratio on the CEC'2013 benchmark**

1107 This section shows the Peak Ratio obtained on the CEC'2013 benchmark in
 1108 the 5 accuracy levels by:

- 1109 • RMAwA (Table 14)
- 1110 • CMA-ES (Table 15)
- 1111 • DE/nrand/2 (Table 16)
- 1112 • dADE/nrand/1/bin (Table 17).
- 1113 • PNA-NSGAI (Table 18)
- 1114 • NVMO (Table 19)
- 1115 • NEA2 (Table 20)

Table 14: Results with RMAwA

Pb	Fun	Dim	Accuracy level				
			$1E-1$	$1E-2$	$1E-3$	$1E-4$	$1E-5$
F_1	f_1	1	1.000	1.000	1.000	1.000	1.000
F_2	f_2	1	1.000	1.000	1.000	1.000	1.000
F_3	f_3	1	1.000	1.000	1.000	1.000	1.000
F_4	f_4	2	1.000	1.000	1.000	1.000	1.000
F_5	f_5	2	1.000	1.000	1.000	1.000	1.000
F_6	f_6	2	0.992	0.992	0.992	0.992	0.000
F_7	f_7	2	1.000	0.920	0.917	0.917	0.917
F_8	f_6	3	0.824	0.824	0.824	0.824	0.824
F_9	f_7	3	1.000	0.519	0.515	0.514	0.513
F_{10}	f_8	2	1.000	1.000	1.000	1.000	1.000
F_{11}	f_9	2	1.000	1.000	1.000	1.000	1.000
F_{12}	f_{10}	2	1.000	1.000	1.000	1.000	1.000
F_{13}	f_{11}	2	0.997	0.997	0.997	0.997	0.997
F_{14}	f_{11}	3	0.823	0.813	0.813	0.813	0.813
F_{15}	f_{12}	3	0.705	0.703	0.703	0.703	0.703
F_{16}	f_{11}	5	0.683	0.670	0.670	0.670	0.670
F_{17}	f_{12}	5	0.668	0.660	0.660	0.660	0.660
F_{18}	f_{11}	10	0.377	0.237	0.237	0.233	0.233
F_{19}	f_{12}	10	0.128	0.128	0.128	0.128	0.128
F_{20}	f_{12}	20	0.253	0.125	0.125	0.125	0.125

Table 15: Results with CMA-ES

Pb	Fun	Dim	Accuracy level				
			$1E-1$	$1E-2$	$1E-3$	$1E-4$	$1E-5$
F_1	f_1	1	1.000	1.000	1.000	1.000	1.000
F_2	f_2	1	1.000	1.000	1.000	1.000	1.000
F_3	f_3	1	1.000	1.000	1.000	1.000	1.000
F_4	f_4	2	1.000	1.000	1.000	1.000	0.990
F_5	f_5	2	1.000	1.000	1.000	1.000	1.000
F_6	f_6	2	0.783	0.783	0.782	0.776	0.000
F_7	f_7	2	0.531	0.529	0.521	0.518	0.516
F_8	f_6	3	0.115	0.115	0.115	0.115	0.115
F_9	f_7	3	0.282	0.278	0.274	0.273	0.272
F_{10}	f_8	2	1.000	1.000	0.998	0.992	0.978
F_{11}	f_9	2	0.990	0.977	0.970	0.963	0.953
F_{12}	f_{10}	2	0.788	0.788	0.778	0.760	0.760
F_{13}	f_{11}	2	0.980	0.967	0.957	0.950	0.947
F_{14}	f_{11}	3	0.760	0.750	0.743	0.743	0.743
F_{15}	f_{12}	3	0.680	0.658	0.655	0.655	0.653
F_{16}	f_{11}	5	0.667	0.667	0.667	0.667	0.663
F_{17}	f_{12}	5	0.585	0.585	0.585	0.585	0.583
F_{18}	f_{11}	10	0.340	0.340	0.340	0.340	0.340
F_{19}	f_{12}	10	0.597	0.597	0.597	0.597	0.597
F_{20}	f_{12}	20	0.448	0.448	0.448	0.448	0.425

Table 16: Results with DE/nrand/2

Pb	Fun	Dim	Accuracy level				
			$1E-1$	$1E-2$	$1E-3$	$1E-4$	$1E-5$
F_1	f_1	1	1.000	1.000	1.000	1.000	1.000
F_2	f_2	1	1.000	1.000	1.000	1.000	1.000
F_3	f_3	1	1.000	1.000	1.000	1.000	1.000
F_4	f_4	2	1.000	1.000	1.000	1.000	1.000
F_5	f_5	2	1.000	1.000	1.000	1.000	1.000
F_6	f_6	2	0.669	0.669	0.669	0.669	0.000
F_7	f_7	2	0.276	0.276	0.276	0.276	0.275
F_8	f_6	3	0.365	0.365	0.365	0.365	0.363
F_9	f_7	3	0.066	0.066	0.066	0.066	0.065
F_{10}	f_8	2	1.000	1.000	1.000	1.000	1.000
F_{11}	f_9	2	0.667	0.667	0.667	0.667	0.667
F_{12}	f_{10}	2	0.635	0.628	0.628	0.618	0.618
F_{13}	f_{11}	2	0.667	0.667	0.667	0.667	0.667
F_{14}	f_{11}	3	0.667	0.667	0.667	0.667	0.667
F_{15}	f_{12}	3	0.413	0.408	0.405	0.400	0.400
F_{16}	f_{11}	5	0.667	0.667	0.667	0.667	0.667
F_{17}	f_{12}	5	0.288	0.283	0.283	0.280	0.280
F_{18}	f_{11}	10	0.517	0.513	0.507	0.507	0.507
F_{19}	f_{12}	10	0.230	0.218	0.203	0.190	0.180
F_{20}	f_{12}	20	0.230	0.230	0.230	0.230	0.230

Table 17: Results with dADE/nrand/1/bin

Pb	Fun	Dim	Accuracy level				
			$1E-1$	$1E-2$	$1E-3$	$1E-4$	$1E-5$
F_1	f_1	1	1.000	1.000	1.000	1.000	1.000
F_2	f_2	1	1.000	1.000	1.000	1.000	1.000
F_3	f_3	1	1.000	1.000	1.000	1.000	1.000
F_4	f_4	2	1.000	1.000	1.000	1.000	1.000
F_5	f_5	2	1.000	1.000	1.000	1.000	1.000
F_6	f_6	2	1.000	1.000	1.000	0.988	0.000
F_7	f_7	2	1.000	0.960	0.878	0.808	0.714
F_8	f_6	3	0.990	0.991	0.985	0.958	0.947
F_9	f_7	3	0.829	0.592	0.552	0.436	0.349
F_{10}	f_8	2	1.000	1.000	1.000	1.000	1.000
F_{11}	f_9	2	0.867	0.667	0.667	0.667	0.667
F_{12}	f_{10}	2	0.750	0.748	0.738	0.740	0.735
F_{13}	f_{11}	2	0.737	0.667	0.667	0.667	0.667
F_{14}	f_{11}	3	0.943	0.667	0.667	0.667	0.667
F_{15}	f_{12}	3	1.000	0.643	0.623	0.600	0.620
F_{16}	f_{11}	5	0.890	0.667	0.667	0.667	0.667
F_{17}	f_{12}	5	0.963	0.480	0.420	0.400	0.358
F_{18}	f_{11}	10	0.663	0.630	0.630	0.613	0.603
F_{19}	f_{12}	10	0.495	0.118	0.080	0.020	0.000
F_{20}	f_{12}	20	0.080	0.005	0.000	0.000	0.000

Table 18: Results with PNA-NSGA

Pb	Fun	Dim	Accuracy level				
			$1E-1$	$1E-2$	$1E-3$	$1E-4$	$1E-5$
F_1	f_1	1	1.000	1.000	1.000	1.000	1.000
F_2	f_2	1	1.000	1.000	1.000	1.000	1.000
F_3	f_3	1	1.000	1.000	1.000	1.000	1.000
F_4	f_4	2	1.000	1.000	0.995	0.985	0.805
F_5	f_5	2	1.000	1.000	1.000	1.000	1.000
F_6	f_6	2	0.562	0.536	0.523	0.473	0.000
F_7	f_7	2	1.000	0.741	0.726	0.709	0.683
F_8	f_6	3	0.352	0.330	0.310	0.275	0.252
F_9	f_7	3	0.480	0.326	0.318	0.298	0.276
F_{10}	f_8	2	1.000	1.000	1.000	1.000	1.000
F_{11}	f_9	2	0.877	0.677	0.670	0.680	0.663
F_{12}	f_{10}	2	0.752	0.715	0.672	0.642	0.573
F_{13}	f_{11}	2	0.697	0.667	0.667	0.663	0.623
F_{14}	f_{11}	3	0.933	0.667	0.667	0.663	0.610
F_{15}	f_{12}	3	0.665	0.495	0.485	0.470	0.443
F_{16}	f_{11}	5	1.000	0.523	0.523	0.417	0.323
F_{17}	f_{12}	5	0.917	0.347	0.338	0.300	0.245
F_{18}	f_{11}	10	0.640	0.117	0.113	0.110	0.093
F_{19}	f_{12}	10	0.020	0.020	0.043	0.017	0.010

Table 19: Results with NVMO

Pb	Fun	Dim	Accuracy level				
			$1E-1$	$1E-2$	$1E-3$	$1E-4$	$1E-5$
F_1	f_1	1	1.000	1.000	1.000	1.000	1.000
F_2	f_2	1	1.000	1.000	1.000	1.000	1.000
F_3	f_3	1	1.000	1.000	1.000	1.000	1.000
F_4	f_4	2	1.000	1.000	1.000	1.000	1.000
F_5	f_5	2	1.000	1.000	1.000	1.000	1.000
F_6	f_6	2	1.000	0.996	0.944	0.681	0.000
F_7	f_7	2	1.000	1.000	0.953	0.901	0.804
F_8	f_6	3	0.411	0.300	0.276	0.198	0.027
F_9	f_7	3	1.000	0.686	0.409	0.279	0.194
F_{10}	f_8	2	1.000	1.000	1.000	1.000	0.967
F_{11}	f_9	2	1.000	0.667	0.667	0.667	0.667
F_{12}	f_{10}	2	0.838	0.743	0.730	0.705	0.593
F_{13}	f_{11}	2	0.997	0.667	0.667	0.667	0.663
F_{14}	f_{11}	3	1.000	0.667	0.667	0.667	0.627
F_{15}	f_{12}	3	1.000	0.723	0.675	0.640	0.378
F_{16}	f_{11}	5	1.000	0.673	0.663	0.663	0.653
F_{17}	f_{12}	5	1.000	0.483	0.453	0.438	0.325
F_{18}	f_{11}	10	0.997	0.470	0.460	0.460	0.327
F_{19}	f_{12}	10	0.273	0.133	0.133	0.127	0.093
F_{20}	f_{12}	20	0.000	0.000	0.000	0.000	0.000

Table 20: Results with NEA2

Pb	Fun	Dim	Accuracy level				
			$1E-1$	$1E-2$	$1E-3$	$1E-4$	$1E-5$
F_1	f_1	1	1.000	1.000	1.000	1.000	1.000
F_2	f_2	1	1.000	1.000	1.000	1.000	1.000
F_3	f_3	1	1.000	1.000	1.000	1.000	1.000
F_4	f_4	2	1.000	1.000	1.000	1.000	0.990
F_5	f_5	2	1.000	1.000	1.000	1.000	1.000
F_6	f_6	2	0.963	0.963	0.958	0.950	0.000
F_7	f_7	2	0.946	0.925	0.918	0.914	0.911
F_8	f_6	3	0.241	0.240	0.240	0.240	0.239
F_9	f_7	3	0.622	0.595	0.584	0.581	0.579
F_{10}	f_8	2	1.000	1.000	1.000	0.988	0.980
F_{11}	f_9	2	0.980	0.967	0.967	0.960	0.960
F_{12}	f_{10}	2	0.853	0.850	0.843	0.840	0.833
F_{13}	f_{11}	2	0.977	0.970	0.960	0.957	0.947
F_{14}	f_{11}	3	0.830	0.817	0.810	0.807	0.800
F_{15}	f_{12}	3	0.743	0.723	0.720	0.718	0.713
F_{16}	f_{11}	5	0.673	0.673	0.673	0.673	0.673
F_{17}	f_{12}	5	0.695	0.695	0.695	0.695	0.695
F_{18}	f_{11}	10	0.667	0.667	0.667	0.667	0.663
F_{19}	f_{12}	10	0.667	0.667	0.667	0.667	0.667
F_{20}	f_{12}	20	0.363	0.360	0.360	0.360	0.350