Multimodal Optimization with the Local Optimum Ranking 2 Algorithm

Francisco Daniel Filip Duarte (danielduarte1181@gmail.com)
https://orcid.org/0000-0003-0523-161X

Research Article

Keywords: multimodal optimization, genetic algorithm, particle swarm optimization, structural optimization, local ranking, metamodel segmentation

Posted Date: October 22nd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-973713/v2

Title: Multimodal Optimization with the Local Optimum Ranking 2 Algorithm

Name: Daniel Duarte
Email: danielduarte1181@gmail.com
Orcid: https://orcid.org/0000-0003-0523-161X
Degree: Master in Applied Science in Mechanical Engineering from Concordia University, Canada (2019)
Location: Oaxaca, Mexico
email to be displayed: danielduarte1181@gmail.com

All figures except fig 3 to be displayed in color.

All references mentioned in the Reference List are cited in the text, and vice versa.

The local optimum ranking 2 (LOR2) is a patent pending algorithm. In order to use it for commercial purposes contact danielduarte1181@gmail.com.
.

Highlights:

1. Innovative Local optimum ranking 2 (LOR2), a new niching algorithm.
2. Provides a local rank, given by the objective value within each local optimum.
3. Identification of local apices, to speeds up design process.
4. A multi-focus, multimodal optimization with equalized exploration effort on each local optimum.
5. Allows to discretize the domain for a thorough function exploration.

Abbreviations:

LOR2 – Local optimum ranking 2

CS – Cuckoo search algorithm

RTC – Crowding with restricted tournament selection

DC – Deterministic crowding

C – Clearing method

FS – Fitness sharing method

L2PS – Local optimum ranking 2 population sorting

RS - Response surface

PV – Polarization vertex

PME - Polarized mesh element

GA-L2PS - Genetic algorithm with L2PS

L2GA - LOR2 genetic algorithm

L2GA-L2PS - LOR2 genetic algorithm with L2PS

GA-C – Genetic algorithm with clearing

GA-RTS – Genetic algorithm with restricted tournament selection

GA-DC – Genetic algorithm with deterministic crowding

L2PSO – LOR2 particle swarm optimization

Multimodal Optimization with the Local Optimum Ranking 2 Algorithm

F. D. Filip Duarte [1],
danielduarte1181@gmail.com, danielduarte@vmxsoftwares.com

[1]Independent - www.filipduarte.com, www.vmxsoftwares.com

ABSTRACT: **In optimization tasks, it is interesting to achieve a set of efficient solutions instead of one single output, in the case the best solution is not suitable. Many niching methods offer a diversified response, yet some important problems are common: (1) The most interesting solutions of each local optimum are not identified. Thus, the output is the overall population of solutions, which increases the work of the designer in verifying which solution is the most interesting. (2) Existing niching algorithms tend to distribute the solutions on the most promising regions, over-populating some local optima and sub-populating others, what leads to a poor optimization.**

To solve these challenges, a novel niching method is presented, named local optimum ranking 2. This sorting methodology favors the exploration of a defined number of local optima, and ranks each local population by objective value within each local optimum. Thus, is performed a multi-focus exploration, with an equalized number of solutions on each local optimum, while identifying which solutions are the local apices. Experimental results demonstrate the local optimum ranking 2 provides superior performance than other popular niching methods, for the selected test functions and global optimization algorithms. Also, its versatility is demonstrated in the several ways it can be combined with some of the most well-known methods.

In a second experiment, the LOR2 algorithm is applied in the design optimization of a metallic cantilever beam. It is exemplified how the LOR2 algorithm can achieve a set of efficient and diverse design configurations, identifying which are the apices of each local optimum. Thus, the LOR2 facilitates multimodal optimization tasks, while offering both performance and diversity for design challenges.

In addition, a third experiment describes how the algorithm can be applied to segment the domain of any function, with any type of input distribution or number of coordinates, into a mesh of similar sized or custom sized elements. Thus, it can segment a response surface named Kriging, significantly simplifying it and reducing computation time.

Keywords: multimodal optimization, genetic algorithm, particle swarm optimization, structural optimization, local ranking, metamodel segmentation.

1. Introduction

The design of many engineering systems is an example of application of some of the most known global optimization algorithms (GOAs). In single-objective optimization (SOO), at each iteration, the population of solutions is ranked by the single objective value, and the best solutions are selected to be part of the next generation of the optimization process, until the algorithm converges, most often to a single point in the domain. Several niching algorithms offer the alternative of providing a diversified set of distinct and competitive responses, in what is also called a multimodal optimization (MMO).

The local optimum ranking 2 (LOR2) provides a novel approach among niching algorithms, in ranking a population of solutions inside each local optimum and downgrading duplicates, allowing to identify and properly process the most competitive local optima of the design space in a multi-focus approach. By downgrading replicate solutions, it allows also have less local conglomeration, facilitating a broader exploration inside each local optimum. The LOR2 provides several metrics for the population of solutions, such as the local optimum each solution belongs to, the rank of each solution according its objective value inside its local optimum, the level of redundancy of each solution, and which are the apices of each local optimum region. Thus, the LOR2 can be combined with GOAs in multiple ways. Besides substituting regular objective value sorting by the local rank sorting, cross-over operations can be applied only to solutions of the same local optimum, the local apices can substitute the global best vector in swarm methods, creating multiple local swarms, and many other combinations can be generated.

In optimization processes, expensive functions often have their output evaluated more frequently at the most promising regions of the domain, rather than in an overall Cartesian grid mapping. Thus, function evaluations occur most often inside a promising local optimum, or a set of local optima if the optimization is a multimodal optimization (MMO). Metamodels become an interesting option in many cases to reduce computation costs, in providing a fast approximation of the response to dismiss unpromising candidate solutions, and provide promising local optimum guesses to speed up optimization. Yet, to achieve response surfaces which can provide greater precision with low computation cost remains as an important challenge. Several response surface models are available in public literature, such as polynomial regression (PR) [1], Kriging (KR) [2], radial basis functions (RBF) [2] among others, which are popular response surfaces applied to speed up optimization processes. Yet their reduced training times and increased accuracy often are object of study.

The local optimum LOR2 also can be applied also to segment the domain of a function, or a response surface, with any number of coordinates into many elements of similar or custom size.

2. Related work

Mathematical or numerical optimization has started at the same time as the advent of Newton's classic mechanics and calculus, with the use of numerical optimization algorithms based on gradient. These methods have been for centuries the core of numerical optimization with application in several fields of engineering. Since the last few decades, several heuristic optimization algorithms have been created. A subgroup of these heuristics is called GOA, which works with a population of solutions. Global optimization is presently applied in many areas of engineering, especially those where a small percentage of efficiency increase can represent the economy of large financial amounts. This is the case of artificial intelligence (AI), logistics, telecommunication, and systems in space, aerospace, automotive, and energy sectors, among others, with many publications available in the public domain.

The genetic algorithm (GA) is a very popular global optimization algorithm (GOA), which is inspired by the theory of evolution of living organisms which reproduce by mating. It states that beneficial mutations tend to accumulate along time, thus generating its evolution. Goldberg [3] describes

the functionality of the GA. Kennedy [4] described the method called particle swarm optimization (PSO), which is based on the flocking movement of birds and the schooling of fishes. Other GOAs which are also popular, are differential evolution (DE) [5], simulated annealing (SA) [6], and cuckoo search (CS) algorithm [7], among many others.

In order to find multiple feasible solutions of a function, multimodal optimization (MMO) algorithms or niching methods were created, which work in conjunction with GOAs. Some well-known niching algorithms are presented:

- **Fitness sharing (FS)** – Created by Holland [8] and improved by Goldberg and Richardson [9], this algorithm mimics how living species compete for resources on nature, which favors its broader distribution on a given geographic region. In this algorithm, the fitness value is divided by a function that integrates the presence of other solutions within a niching radius.
- **Clearing method (C)**- In this MMO algorithm, a number of the best solutions within a niching radius have their fitness preserved. Exceeding solutions have their fitness cleared.
- **Crowding with redistricted tournament selection (RTS)** – Created by Harik in 1995 [10], in this method, a GOA has a population POP_P which generates POP_Q. A set of N solutions randomly chosen from POP_P is selected. A single solution of POP_Q competes with the closer solution of the set, and the best solution is kept in the next generation. This process is repeated to all solutions of POP_Q.
- **Deterministic Crowding (DC)** - In a GOA which performs crossover operations, each offspring of the generated pair competes with the closest parent. The solution with the best fitness is kept for the next generation. This method was created in 1992 by Mahfoud [11]

De Jong introduced the concept of crowding methods [12] in 1975. Among other niching methods are derating [13], parallelization [14], and clustering [15].

The Institute of Electrical and Electronics Engineers (IEEE) presents the world congress of computational intelligence (WCCI) [16], providing conferences on neural networks, fuzzy systems, and evolutionary computation. The congress of evolutionary computation (CEC) promotes a niching competition every year [17], where participants evaluate the efficiency of their niching algorithms by benchmarking it with a test work function suite [18].

Li et al. published in 2017 a survey on the latest progress of MMO algorithms [19], listing several of the most popular methods and some recent developments. They describe the operation of several niching methods like FS, crowding variations, DC, derating, among others. Also, they mention that other non-evolutionary algorithms, like PSO and DE, have been applied with a niching approach, and expound the challenges commonly faced in using niching algorithms while testing it on benchmark functions.

Sareni and Krahenbuhl [20] describe the operation of several niching algorithms, including FS, four types of crowding and clearing, and exemplify their application in test problems. Passaro and Starite propose a niching PSO [21], exploring several variations. Wong provides a short survey on MMO algorithms [22].

Yue et al. [23] propose a DE with crowding distance for MMO. Liang et al. [24] describe a clustering-based DE for MMO. Wu et al. [25] applied a multimodal ant colony optimization for imaging sensors. Polakova [26] et al. describes a DE which preserves diversity in optimization.

Truss structures are interesting numerical models to test the performance of optimization algorithms. While it has a simple formulation, it illustrates well the challenges faced in the optimization of structural designs and other domains, like the presence of several local optima. Deb and Surendra applied GA for the weight reduction of 2D and 3D trusses [27], by seeking optimal topology and cross section sizes. Deb also describe the optimization of design parameters of a class of welded structures, using GA and proving it has better performance than other tested methods [28]. Gomes and Beck [29] describe the use of the PSO algorithm in the optimization of a steel-frame transmission line tower, with the aid of a neural network as metamodel. Wang et al. [30] describe the optimization of truss structures with nodes position displacement and cross-section variation, considering frequency constraints, and describe the challenges faced in not getting trapped in a local optimum response. Kaveh [31]–[34] also applies several heuristic algorithms like ant colony, PSO, among others, in the optimization of truss structures. Azad and Hasançebi [35] propose a self-adaptive algorithm in the optimization of trusses, and Lingyun et al. [36] apply GA to optimize the shape and size of trusses with frequency constraints. Gomes [37], Perez and Behdinan [38] also optimize a truss, applying a particle swarm algorithm. Cheng [39] applied GA in the design of steel truss arch bridges.

In the field of MMO applied to the design of structures, interestingly there are fewer examples. Lin and Chen propose a multistage algorithm for multimodal optimization of structures [40]. Islam proposes a niching evolutionary algorithm to find multiple solutions of a truss structure [41]. Huang et al. propose a niching PSO [42] for the design of composite structures. Other publications also present the MMO applied to structures [43]–[45], in dealing with the challenge of providing a diverse set of efficient designs.

There are a large number of publications in SOO of structures, mainly related to industrial research and development in the field of automotive, civil engineering, space, and aerospace sectors. In order to evaluate the efficiency of more complex structural designs, finite element models (FEM) are one of the most popular type of mathematical models. This method segments the structure in several elements, and their interaction is evaluated generating a field of deformation and stress. Cook et al. offer a reference with the fundaments of FEM [46]. Ugural [47] also offers an introduction of analytical stress analysis of structures, and FEM formulation.

Liu and Tovar [48] generated a topology optimization study in MATLAB, to reduce weight of structures. Gauchia et al. [49] performed the torsional stiffness and weight optimization of a bus structure using GA and FEM. Park and Dang [50] applied a CAD-CAE interface integration to automate the design optimization of metallic structural components, using GA and supported with metamodel to reduce computation costs. Zheng et al. [51] designed a lunar lander using FEM, using commercial software called Abaqus and LS-Dyna. These are only a few of many examples of structural design studies applying FEM and optimization algorithms.

In optimization, a metamodel, which is also called a response surface or surrogate model, can be applied to reduce computation costs. This is achieved by dismissing unpromising candidate solutions and suggesting promising alternatives. Among well-known response surface Kriging (KR) also called Gaussian process [52], polynomial regression [1], radial basis functions [2], and ANNs [53]. To evaluate the efficiency of a response surface in forecasting the output of a function, many benchmark functions were created [2].

In the design of very efficient structures, GOAs, gradient-based, and other optimization algorithms are applied in conjunction with mathematical models of the structures. Since the computation costs of the structural models are often expensive, metamodels are often applied. If the structural model is an aeronautical or spatial structure, aerodynamics models, which also are computationally expensive, are added to the structural model in a so-called multidisciplinary design optimization (MDO).

In the field of AI, Rumelhart et al. described the backpropagation method [53], which is applied to train multi-level ANNs. These ANNs are a type of metamodel, which consists of a mathematical model which mimics the flow of information observed in organic neuronal tissues of several living species, which allows many living organisms to have intelligence. This methodology has been very efficient in text, image, and speech recognition, among other applications. Among some of its generalized applications, ANNs can be applied to forecast the response of a mathematical function, what is called regression, and for category classification.

Deep learning (DL) [54], is a technique created by Lecun, Bengio, and Hinton, where many layers of ANNs are trained over data set to recognize several levels of patterns. It has unsupervised training, different than ANNs, and has many important applications for the automation of many complex

and challenging tasks, substituting human work in many fields of activity with similar or greater levels of intelligence. DL and other areas of AI have as their basic building blocks response surfaces such as ANNs, KR among others metamodels. Since usual AI tasks involve large data sets, large metamodels are required, which require elevated training times. This demands large parallel computing infrastructure for their often computationally very expensive metamodel training procedures.

3. Methodology

Several niching algorithms provide options to explore both efficiency and diversity of a function. However, as mentioned, these algorithms have some important hindrances: (1) they provide as output the overall population, not identifying which solutions are the local apices, giving a large workload to designers to verify, often manually, which solution is the most suitable for the design task. Also, (2) most niching algorithms distribute the population of solutions on the locations of the domain which presented best fitness, what can overpopulate some local optima and reduce the number of solutions exploring other promising local optima. Since a local optimum can present a poor outcome in the initial phase of the exploration, yet a competitive output if properly processed, these niching algorithms can present low performance for several functions.

The LOR2 keeps in the optimization process all best-evaluated regions of interest of the design space, with an equalized number of solutions on each local optimum to properly process it. Also, it downgrades solutions considered redundant, thus a broader exploration is achieved inside each local region. In addition, it provides several metrics for the population of solutions, which allows it to be combined with GOAs in multiple ways: the rank of each solution inside each local optimum region, its redundancy level, and which solution is the local apice.

The LOR2 evaluates the solutions of a population-based GOA, in order to explore the most feasible regions of the design space. For this, the design space is normalized, where each design variable is defined between zero and one. Once each solution is evaluated by the single objective value, the population of solutions is sorted with the solution with the best objective value at first. With a normalized optimization space, is possible to calculate the distance matrix $\boldsymbol{D}$ between each solution of the sorted population. Assuming a normalized optimization space with H dimensions can be represented by a hypercube, the size of the greatest diagonal inside this hypercube L is given as the square root of H. To have all distances in the design space between 0 and 1, the normalized distance matrix $\boldsymbol{Z}$ is defined as the distance matrix $\boldsymbol{D}$ divided by the size of the greatest diagonal of the design space hyper-cube L.

In the LOR2 the local optimum apice is a solution with the best objective value within the local optimum, and a sub solution is a point belonging to a local optimum that is not a leader. Each local optimum leader receives a local optimum rank equal to zero, and each sub solution receives the local optimum rank equal to the number of solutions non-penalized for redundancy in the same local optimum, which have better objective value. Four operational variables are defined: d_1 which is the local optimum radius, d_2 which is the redundancy radius, the number of allowed replicates N_r, and the number of allowed local apices N_l.

If a solution does not have another solution with a better objective value within a distance smaller than d_1, it can potentially be an apice. If the number of local apices already identified by the algorithm is lower than the maximum number of allowed apices N_l, then this solution is also considered as an apex. Since there is a limited number of allowed apices, exceeding leader candidates are defined as a sub solution of the closest apex recorded by the algorithm.

Following this, it is necessary to define the redundancy penalty of each solution. If two solutions have their distance smaller than the redundancy radius d_2, then the solution with a worse objective value has its redundancy penalty increased by one. In order to allow intensive localized exploration, a number of allowed replicates, N_r, are allowed to each solution and are not penalized for redundancy.

Once each point has its redundancy penalty and local optimum rank evaluated, a rank vector with three variables is generated, with the following sequence: redundancy penalty, local optimum rank, and objective value. This will prioritize non-redundant solutions in competitive regions of the design space, with the best objective values, promoting an equal number of solutions on each local optimum. With this, in the LOR2 sorting, the candidate solutions are sorted by the first rank variable, and then for identical values in the first variable, the second variable is the sorting parameter. The same rule is applied for the third rank variable, and thus the sorting process is accomplished.

Figure 1 displays a curve with five local optima in a 2D domain.

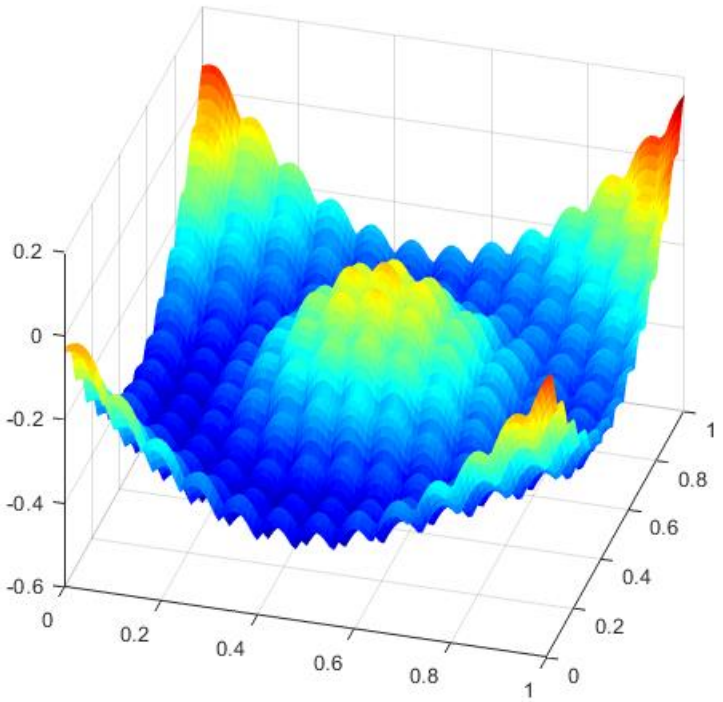

Fig. 1 – A curve denominated "5 local optima with noise"

To illustrate the advantages of the LOR2 algorithm, its operation is compared with other methods in figure 1. The genetic algorithm with hill climb (GA+HC), simulated annealing with hill climb (SA+HC), and the genetic algorithm with hill climb and local optimum ranking 2 (LOR2+GA+HC) are applied to optimize the function with 5 local optima (fig.1), and the distribution of the solutions in the domain at iterations 2, 4 and 6 are presented

at figure 2. It is possible to note that the LOR2 algorithm favors a multi-focus exploration of the domain while keeping local conglomeration under adjusted levels. In contrast, the other algorithms provide an overall exploration of the domain and then focus on a single region, converging all populations of candidate solutions to one single point.

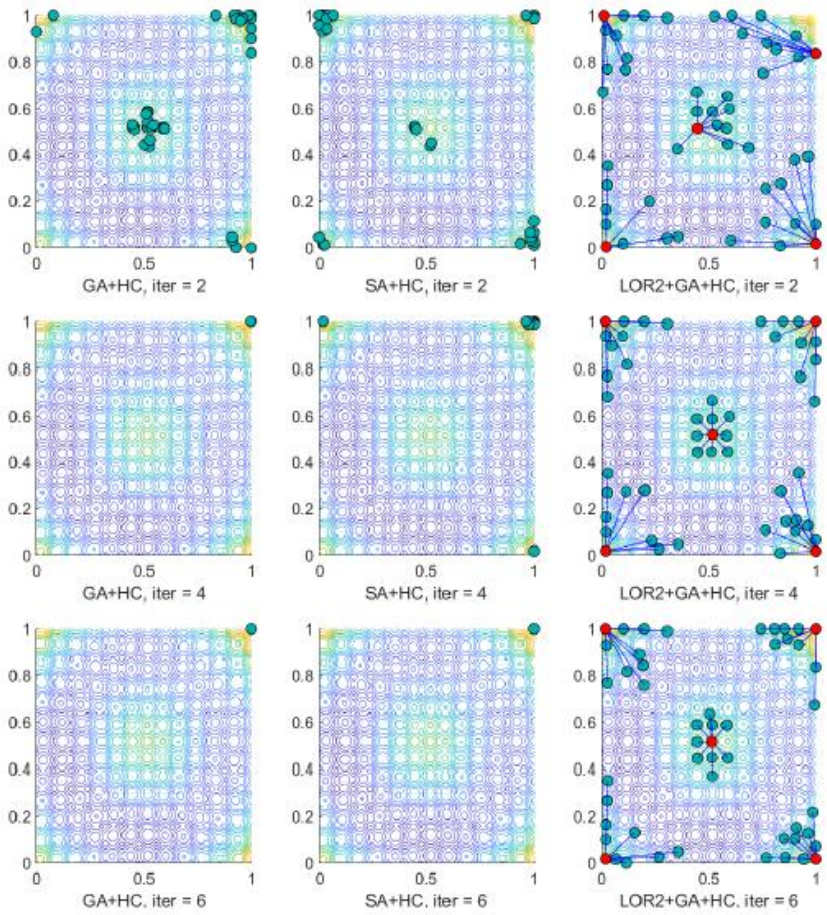

Fig. 2 - Example of the LOR2 multi-focus approach in the optimization of a function with 5 local optima

The LOR2 is presented for a minimization task, at algorithms 1 and 2.

Algorithm 1: *Local Optimum Ranking 2 – Phase 1 (LOR2-1)*

```
1:   procedure LOR2-1(Pop,iter,par )
2:   Sort the population of solutions Pop by objective value,
3:   with best objective value at first
4:   Calculate the normalized distance matrix Z of sorted Pop
5:   ApicesCount ← 0
6:   ApicesList  ← []
7:   Reset all LOR2 variables
8:   for i = 1 to par.PopSize do
9:      PointDefined ← 0
10:    (verify if the point is close to a recorded local optimum apex)
11:    for j = ApicesList do
12:       if Z(i,j) <= par.d1 do
13:          PointDefined ← 1
14:          Pop(i).LocalOptimum ← Pop(j).LocalOptimum
15:          Pop(i).IsLocalApice ← 0
16:          break for loop
17:       end if
18:    end for
19:    (if it is not close to a local optimum then it can be a local optimum apice)
20:    if PointDefined = 0 do
21:       if ApicesCount < par.MaxApicesCount do
22:          ApicesCount ← ApicesCount + 1
23:          ApicesList ← [ApiceList, i]
24:          Pop(i).LocalOptimum = ApicesCount
25:          Pop(i).IsLocalApice ← 1
26:       (if local optimum apices list is full, select closer leader)
27:       else if ApicesCount >= par.MaxApicesCount do
28:          SmallerDistance ← inf
29:          for j = ApicesList  do
30:             if Z(i,j) < SmallerDistance  do
31:                CloserLocalApice ← j
32:                SmallerDistance ← Z(i,j)
33:             end if
34:          end for
35:          Pop(i).LocalOptimum ← Pop(CloserLocalApice).Local optimum
36:          Pop(i).IsLocalApice ← 0
37:       end if
38:    end if
39:  end for
```

As a variation, line 11 of Algorithm 1 can be substituted by:

```
for j = i-1 do
```

This will generate the local optimum ranking 2 with dominance propagation (LOR2-DP), where the local dominance is propagated to neighboring solutions with worse objective value, throughout the design space. This variation has proven to be more efficient for algorithms which perform permutation such as GA, and a worse for combination with swarm methods such as PSO. One great advantage of this variation is that is can identify all local optima of a function, withtin a noise radius of d_1, if the number of apices is set to infinite.

With this, it is necessary to evaluate the redundancy penalty and the local optimum rank of each point, as described at algorithm 2, of LOR2 phase 2:

Algorithm 2: *Local Optimum Ranking 2 – Phase 2 (LOR2-2)*

```
1:   procedure LOR2-2(Pop,iter,par )
2:   (define redundancy level of each solution)
3:   for i = 1 to par.PopSize do
4:      for j = 1 to i-1 do
5:         if Z(i,j) < par.d2 do
6:            Pop(j).ReplicasQty ← Pop(j).ReplicasQty + 1
7:            DeltaRedundacy ← Pop(j).ReplicasQty – par.AllowedReplicates
8:            Pop(i).RedundacyPenalty ← max(0,DeltaRedundacy)
9:            Break loop
10:        end if
11:     end for
12:  end for
13:  RegionCount ← reset
14:  (define the local rank for each solution)
15:  for i = 1 to par.PopSize do
16:     LocalOptimum ← Pop(i).LocalOptimum
17:     Pop(i).LocalRank ← RegionCount(LocalOptimum)
18:        if Pop(i).RedundacyPenalty = 0 do
19:           RegionCount(LocalOptimum) ← RegionCount(LocalOptimum) + 1
20:        end if
21:  end for
22:  for i = 1 to par.PopSize do (write ranks and sort)
23:     Pop(i).Rank(1) ← Pop(i).RedundacyPenalty
24:     Pop(i).Rank(2) ← Pop(i).LocalRank
25:     Pop(i).Rank(3) ← Pop(i).ObjectiValue
26:  end for
27:  Sort Pop by rank 1 to 3 and select the best solution
```

It is important to note that the algorithm is configurated for minimization tasks. For a maximization task, the sorting at line 2 of algorithm 1 should have the greater objective value at first, and the objective value at line 25 of algorithm 2 must be multiplied by -1.

The normalized distance calculation formula is based on the Euclidean distance between two points, divided by the length of the greater diagonal of the normalized design with N dimensions. Thus the $\boldsymbol{Z}$ matrix formulation is defined:

$$Z(i,j) = \frac{Dist(i,j)}{\sqrt{N}} \qquad \text{(eq. 1)}$$

An option to reduce computation costs is to only calculate distances as required by the algorithm and store the calculated values.

Default parameters selection for the LOR2 can follow the values defined in experiment I. In order to customize the operational parameters of the LOR2, it is possible to evaluate the maximum number of possible local optima in the design space, and calculate which fraction of the domain is to be considered in the optimization.

If we call t any positive integer close to x, were:

$$t < x < t + 1$$

We define the floor function as:

$$floor(x) = t \qquad \text{(eq. 2)}$$

In a given space with H dimensions, the number of possible local optima is given by the number of possible hyper-cubical subspaces which can fit in the design space, which is an approximation for the hyper-spherical sub-regions. The number of intersecting local optima that can fit in one dimension, N, is given by the normalized length divided by the radius of the hyper-sphere of the local optimum. Since in the algorithm all distances are divided by the length of the greater diagonal, the radius of the hyper-sphere is multiplied by the square root of H. Thus, we have:

$$N = floor\left(\frac{1}{d_1\sqrt{H}}\right) \qquad \text{(eq. 3)}$$

If we call b the approximate quantity of possible local optima in the overall design space, we have:

$$b = N^H \qquad \text{(eq. 4)}$$

With this, it is convenient to define the local optimum radius d_1, the redundancy radius d_2 and the number of allowed apices N_l in the design space, according to the population size and the fraction of the domain expected to be explored in the optimization. The number of solutions per local optimum N_{sn} must be enough to allow an appropriate local exploration, and is defined by the size of the population of solutions N_p in the overall design space divided by the number of local optima, which is given by the number of allowed apices N_l:

$$N_{sn} = N_p \,/\, N_l \qquad \text{(eq. 5)}$$

It is interesting to assure that on each local optimum, the solutions would not concentrate near the apex, but rather would have a broad distribution, which is achieved with the redundancy penalty. However, if the solutions are too dispersed inside each local optimum, intensive numerical exploration could be compromised.

The LOR2 can be applied to sort a population of solutions favoring an equal distribution on each considered local optimum of the design space. For this, the GOA working with the LOR2 sorting must be operating with aggregated population like occurs for example in the GA. In the GA, the

operators of mutation and crossover are applied to an existing population POP_P, and thus another population, POP_Q, is generated. After POP_Q has its objective values defined, the two populations are added (aggregated), generating POP_R. This population POP_R, in the original configuration of the GA, is then sorted according to the single objective value, and the best solutions are kept, generating POP_P of the following iteration.

The LOR2 can be applied by substituting the regular objective value sorting with the LOR2 sorting, favoring a local optima distribution, as illustrated in figure 3. This sorting with the LOR2 is called local optimum ranking 2 population sorting process (L2PS).

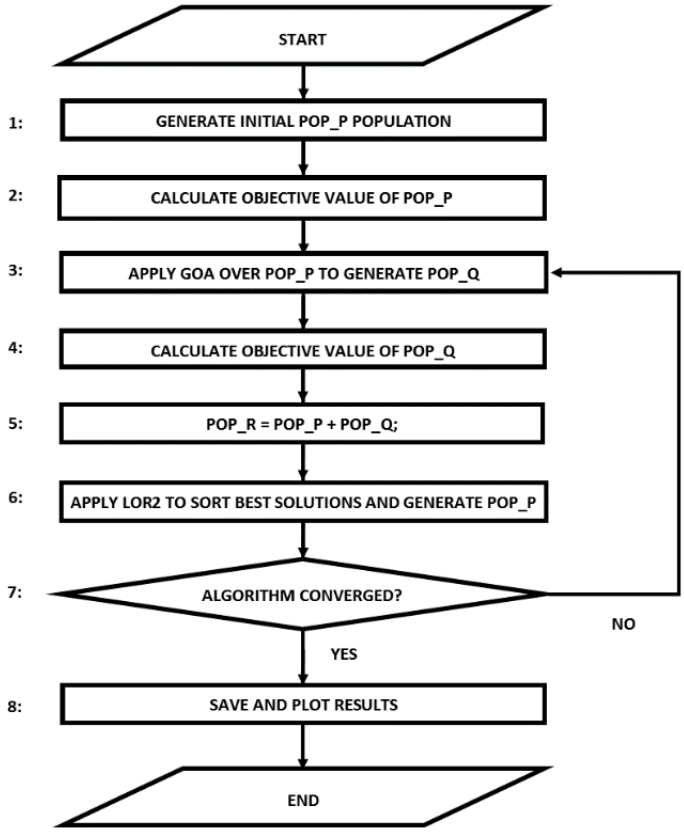

Fig. 3 – Flow chart exemplifying the application of the L2PS process in an optimization algorithm

Some algorithms do not work with aggregated populations, but rather work with ameliorated populations, which is the case of simulated annealing and PSO. In these algorithms, the population POP_P of the following generation is made of some solutions of the existing POP_P population and some solutions of the POP_Q population, by following the particular selection process of each method. This does not allow the L2PS process to be applied, therefore to apply a local rank sorting it is necessary to modify these algorithms to work with aggregated populations. Thus, it is generated a larger POP_R population which can then be sorted by the L2PS, and generate the population POP_P of the following generation with a local optimum ranking distribution.

By allowing large or an unlimited number of local optima, the LOR2 algorithm also can be applied to generate a mesh for a populated domain with any number of coordinates, with a Cartesian, random or clustered distribution. It can generate a mesh for a 2D or 3D finite element model (FEM) [55], or to generate a response surface mesh (RSM) with any number of coordinates, among other applications. The advantages of an RSM is that segmentation allows parallel computing which can reduce training times, precision can be increased with mesh refinement, and individual mesh elements can be trained or updated individually. Also, some models like KR are significantly simplified by having its dataset partitioned, providing its great accuracy with much faster implementation.

In order to provide an efficient curve approximation inside each RSM element, each element must be sufficiently populated to provide gradient in all the dimensions of the function. For a linear approximation, the number of points inside each element needed to generate gradient must be equal to the number of dimensions of the function plus one. If the approximation is of a quadratic polynomial, as in the PR, one more point is needed. Thus, for a polynomial approximation of grade N, applied over a domain of M dimensions, the quantity needed of inputs inside each element is given by the sum of M and N. In order to provide greater precision in regions of interest by mesh refinement, the local optimum radius d_1 can be a function of the position of the local optimum apex in the design space.

Once the population of local optima is defined by the LOR2, it is possible to calculate the position of the polarization vertex (PV) of each element, which is given by the average position of all inputs of the same RSM element. With all PVs defined, each polarized mesh element (PME) is generated by the points of the data set closer to each PV, as in a Dirichlet partition, which is also called Voronoi partition [56].

One challenge arising from generating a mesh over a large data set is the computing cost related to the calculation of distances between each input. A possible alternative is segmenting the data set in regions of similitude, having the care to include adjacent apices of other regions, which is also a viable alternative to generate a 2D or 3D mesh for a FEM. Another alternative is to define the PVs over a sparser data set, and include all the inputs only for the evaluation of each PME function.

At figure 4 at the left side is noted the related local optimum subspaces of the function displayed at figure 1 and its local optimum apices, identified by the LOR2. At the right side at figure 4, its respective PMEs are displayed. These subspaces are defined for a limited number of local apices. The domain is segmented in five main subspaces, related to each local optimum. The local apices at left, and PVs at right, are marked with different colors.

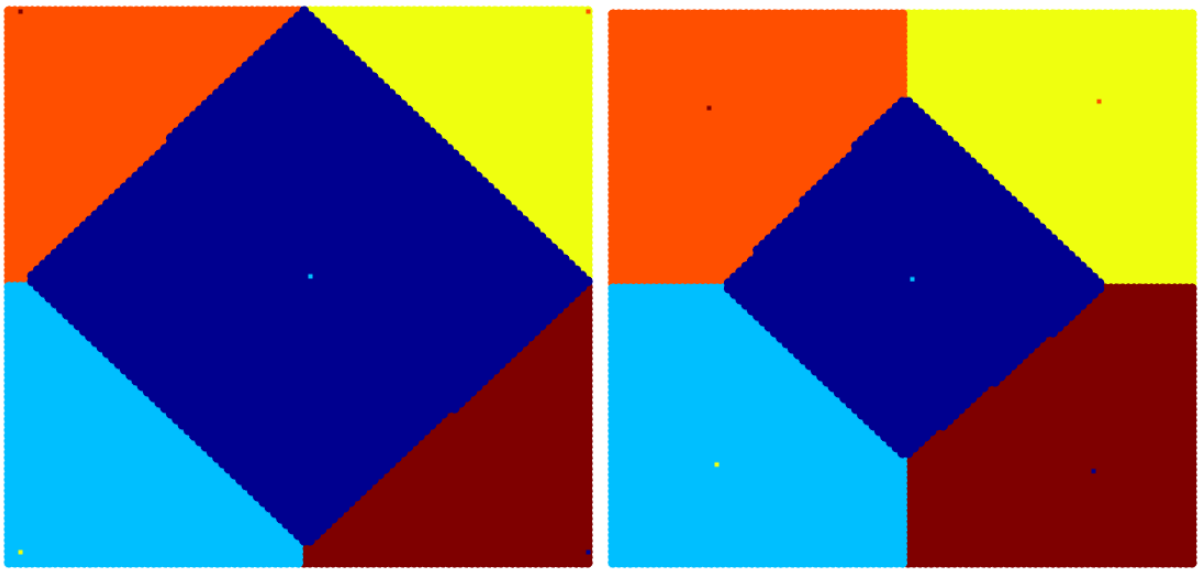

Fig. 4 – Local optimum distribution and respective local apices at the left side. PMEs and respective PVs at right side. LOR2 parameters are local optimum radius d_1=0.1 and number of allowed apices N_l=5.

At figure 5 is demonstrates the difference of operation of the LOR2 by allowing an unlimited number of local optimum apices. At figure 5 at the left side is possible to see the "fish scale" pattern of local optimum distribution. At right side, the PMEs are illustrated. Each local apex and each PV are marked with a point with different color.

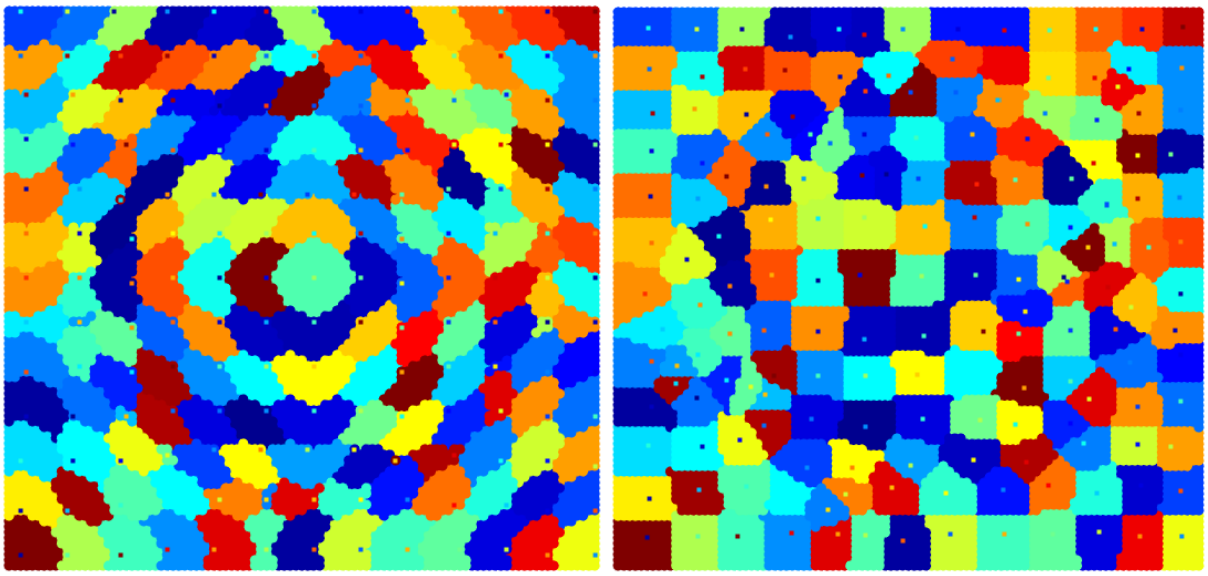

Fig. 5 – Local optimum distribution and respective local apices at the left side. PMEs and respective PVs at right side. LOR2 parameters are local optimum radius d_1=0.05 and number of allowed apices N_l=inf

4. Experiment I: Benchmark Functions Optimization

In order to compare the optimization efficiency of several niching methods with the LOR2, these algorithms are applied to find multiple solutions of a selected test work functions suite. The chose niching methods are:

- **Clearing (C)**
- **Crowding with redistricted tournament selection (RTS)**
- **Deterministic Crowding (DC)**

Two GOAs are selected to be combined with the niching algorithms: GA and PSO. Several combinations are generated from these GOAs and niching methods. The LOR2 can be combined with GA in basically two ways. The first is the simplest, by substituting the regular objective value sorting by the L2PS, generating the GA with L2PS (GA-L2PS). The second way is in performing crossover only with solutions belonging to the same local optimum, generating an LOR2 type of GA (L2GA). A third combination can also be created, by applying both combinations previously described, creating the LOR2 modified GA with L2PS (L2GA-L2PS). Thus, the three combinations are presented:

A. **GA-L2PS** - Genetic algorithm with L2PS
B. **L2GA -** LOR2 genetic algorithm

C. **L2GA-L2PS** - LOR2 genetic algorithm with L2PS

The application of the other niching methods to the GA is direct, giving another three niching algorithms:

D. **GA-C** – Genetic algorithm with clearing
E. **GA-RTS** – Genetic algorithm with restricted tournament selection
F. **GA-DC** – Genetic algorithm with deterministic crowding

In order to combine LOR2 with PSO, the global best vector of the PSO algorithm is substituted by the local apice, identified by the LOR2. Thus, several local swarms are generated, and the number of solutions on each swarm is equalized. Since all other selected niching algorithms do not have an option to deal with the global best vector of the PSO algorithm, and transform the PSO from single modal optimization method to a niching algorithm, only LOR2 is combined with PSO. Thus, we have:

F. **L2PSO** – LOR2 particle swarm optimization

These seven chosen algorithms are applied to selected test work functions from the CEC MMO benchmark suite [18]. Table 1 presents all the 20 functions of the suite.

index		Multimodal test functions	No. global optima
1	F1	Five-Uneven-Peak Trap (1D)	2
2	F2	Equal Maxima (1D)	5
3	F3	Uneven Decreasing Maxima (1D)	1
4	F4	Himmelblau (2D)	4
5	F5	Six-Hump Camel Back (2D)	2
6	F6	Shubert (2D)	18
7	F6	Shubert (3D)	36
8	F7	Vincent (2D)	81
9	F7	Vincent (3D)	216
10	F8	Modified Rastrigin - All Global Optima (2D)	12
11	F9	Composition Function 1 (2D)	6
12	F10	Composition Function 2 (2D)	8
13	F11	Composition Function 3 (2D)	6
14	F11	Composition Function 3 (3D)	6
15	F11	Composition Function 3 (5D)	8
16	F11	Composition Function 3 (10D)	6
17	F12	Composition Function 4 (3D)	8
18	F12	Composition Function 4 (5D)	6
19	F12	Composition Function 4 (10D)	8
20	F12	Composition Function 4 (20D)	8

Table 1 - Test work functions from the CEC MMO benchmark suite

The algorithms are applied to optimize each one of the 20 functions of the suite, 50 times for each function for the 5 levels of precisions, from 1.0 E-1 to 1.0 E-5. From the results of each set of 50 optimization runs, two parameters are defined. The first is called peak ratio (PR), which is given by the number of solutions found by the algorithm on each run, divided by the number of present solutions of the function. The variable NPF_i is the number of peaks found on each run, NKP is the number of known peaks of each function, and NR is the number of runs that is equal to 50.

$$PR = \frac{\sum_{run=1}^{NR} NPF_i}{NKP * NR} \qquad \text{(eq. 6)}$$

SR is the success ratio, which is calculated by the number of successful optimizations runs NSR where all the solutions were found, divided by the number of total runs NR.

$$SR = \frac{NSR}{NR} \qquad \text{(eq. 7)}$$

In order to calculate the number of fitness evaluations per function, $AveFEs$, also called convergence speed, from all the 50 optimization runs, we have:

$$AveFEs = \frac{\sum_{run=1}^{NR} FE_i}{NR} \qquad \text{(eq. 8)}$$

In order to calculate the overall average of the number of fitness evaluation (ANFE), the $AveFEs$ value of each function is summed, and is divided by the number of functions NF.

$$ANFE = \frac{\sum_{run=1}^{NF} AveFEs_i}{NF} \qquad \text{(eq. 9)}$$

To define optimum parameters for the LOR2, functions of table 1 with index 11 and 12 were applied to evaluate several parameter values combinations of d_1 and d_2. By varying d_1 and k from 0.05 to 0.5 in intervals of 0.05, d_2 is defined as the multiplication of k by d_1. The number

of local optimum apices N_l, for the hyper-parameter optimization is defined as 5, and the number of allowed replicates N_r is equal to 4. Each function was optimized 10 times, and the overall value of PR among all functions, for each combination of d_1 and k is presented at table 2. As we can see in the graphic, the optimal configuration for the parameters is 0.20 for d_1 and 0.01 for d_2 for the selected test functions. Further LOR2 operational parameters are presented in table 3.

		d1								
		0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
0.05		0.007	0.013	0.003	**0.023**	0.007	0.013	0.010	0.007	0.013	0.005
0.10		0.003	0.013	0.013	0.007	0.013	0.010	0.005	0.010	0.003	0.013
0.15		0.003	0.010	0.007	0.017	0.010	0.015	0.013	0.010	0.010	0.007
0.20		0.007	0.010	0.007	0.018	0.005	0.007	0.007	0.005	0.015	0.007
0.25		0.007	0.013	0.005	0.000	0.007	0.003	0.005	0.003	0.005	0.013
0.30	k	0.022	0.005	0.005	0.013	0.007	0.013	0.007	0.007	0.010	0.005
0.35		0.007	0.007	0.007	0.007	0.003	0.005	0.010	0.010	0.007	0.010
0.40		0.007	0.007	0.005	0.013	0.010	0.007	0.013	0.007	0.010	0.005
0.45		0.010	0.010	0.013	0.003	0.003	0.003	0.007	0.010	0.003	0.010
0.50		0.010	0.003	0.007	0.005	0.007	0.007	0.007	0.010	0.010	0.010

Table 2 – Average PR for d_1 and k combinations

In order to test the efficiency of the LOR2 before other niching methods on the selected MMO test functions, the optimal parameters of d_1 and d_2 are applied to the algorithms combined with the LOR2. The GOAs combined with LOR2 have the operational parameters named as: number of allowed apices N_l, number of allowed replicates N_r, local optimum radius d_1, and redundancy radius d_2, as presented at table 3. The number apices are close to the number of global optima, thus offering a good exploration of each local optimum of interest. Since for some functions the number of global optima is large, the local optimum radius is reduced to its half, to accommodate all present local optima. For the clearing method, C_k defines the number of the best solutions inside the same region with niching radius r which does not have their fitness cleared. In the RTS method, the number of selected solutions for the tournament is given by $nRTS$. The population size is also defined according to the number of global optima of each function and the size of the design space and is the same value to all algorithms. The seven algorithms were applied to optimize the selected test work functions as previously described. In table 3, the variable nD defines the number of dimensions of each test function, and r is the niching radius. Both values are fixed, defined by the test functions suite.

#	Function	D	r	No. G. Optim	Pop Size	Nl	Nr	d1	d2	Ck	nRTS
1	F1	1	0.01	2	100	5	4	0.200	0.010	3	10
2	F2	1	0.01	5	100	8	4	0.200	0.010	3	10
3	F3	1	0.01	1	100	5	4	0.200	0.010	3	10
4	F4	2	0.01	4	100	5	4	0.200	0.010	3	10
5	F5	2	0.50	2	100	5	4	0.200	0.010	3	10
6	F6	2	0.50	18	480	20	4	0.100	0.010	3	10
7	F7	2	0.20	36	480	40	4	0.100	0.010	3	10
8	F6	3	0.50	81	480	100	4	0.100	0.010	3	10
9	F7	3	0.20	216	1000	200	4	0.100	0.010	3	10
10	F8	2	0.01	12	200	15	4	0.100	0.010	3	10
11	F9	2	0.01	6	200	8	4	0.200	0.010	3	10
12	F10	2	0.01	8	200	10	4	0.200	0.010	3	10
13	F11	2	0.01	6	200	8	4	0.200	0.010	3	10
14	F11	3	0.01	6	200	8	4	0.200	0.010	3	10
15	F12	3	0.01	8	240	10	4	0.200	0.010	3	10
16	F11	5	0.01	6	200	8	4	0.200	0.010	3	10
17	F12	5	0.01	8	240	10	4	0.200	0.010	3	10
18	F11	10	0.01	6	480	8	4	0.200	0.010	3	10
19	F12	10	0.01	8	480	10	4	0.200	0.010	3	10
20	F12	20	0.01	8	480	10	4	0.200	0.010	3	10

Table 3 – Operational parameters for LOR2 and other niching methods

With the following parameters, the 20 selected test work functions are optimized and the detailed values of PR and SR, for the 6 algorithms and the 5 levels of precision, are presented in appendix B. The overall average values are presented in table 4.

Average values of all functions										
Precision	1.0 E-01		1.0 E-02		1.0 E-03		1.0 E-04		1.0 E-05	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	0.399	0.313	0.304	0.241	0.223	0.123	0.124	0.059	0.070	0.051
L2GA-L2PS	0.391	0.285	0.288	0.194	0.195	0.118	0.106	0.066	0.062	0.054
L2GA	0.345	0.249	0.191	0.077	0.140	0.057	0.095	0.036	0.058	0.021
L2PSO	**0.791**	**0.659**	**0.605**	0.296	**0.578**	**0.296**	**0.541**	**0.283**	**0.500**	**0.259**
GA-C	0.672	0.608	0.422	**0.343**	0.300	0.176	0.155	0.065	0.080	0.055
GA-DC	0.365	0.275	0.201	0.093	0.143	0.057	0.104	0.053	0.061	0.029
GA-RTS	0.492	0.347	0.269	0.157	0.172	0.105	0.108	0.065	0.068	0.054

Table 4 – Average values of PR ad SR for each level of accuracy

It is possible to see that the combination of the LOR2 with PSO (L2PSO) achieved the best overall average for PR in all the accuracy levels, and 4 out of 5 best averages for SR values. Thus, the ability of the algorithm to achieve a greater percentage of all local optimum solutions, demonstrates that the multiple local swarms of the L2PSO provide a good overall search in the design space. It proved to achieve also a precise localized exploration, offering good performance for all levels of accuracy. The GA-C provides also good results, being in second place for all other values for PR and SR, proving the clearing method is also a competitive niching algorithm.

Overall Average			
algorithm	PR	SR	ANFE
GA-L2PS	0.224	0.157	242063.0
L2GA-L2PS	0.208	0.143	245366.0
L2GA	0.166	0.088	243990.0
L2PSO	**0.603**	**0.359**	231288.0
GA-C	0.326	0.249	**220934.0**
GA-DC	0.175	0.101	242566.0
GA-RTS	0.222	0.146	242328.0

Table 5 – Average values of PR, SR and ANFE for all levels of accuracy

Table 5 displays the overall average of PR and SR. It shows the calculated value of PR equal to 0.603 and SR as 0.359 for the L2PSO algorithm, which is significantly higher than the algorithm in second place, GA-C. In terms of the average number of fitness evaluations (ANFE), GA-C achieved 220934, with the lowest number, and L2PSO achieved 231288, in second place.

These results illustrate how L2PSO is an interesting and robust alternative to MMO challenges when compared to 3 other niching methods. The good performance of the L2PSO was achieved by using the capacity of high accuracy of the PSO, with the multiple local optimum exploration of the LOR2 in generating multiple local swarms. Since the other niching methods do not offer an alternative in modifying the global best vector of the PSO algorithm, it only was combined with LOR2. Also, the LOR2 provided 3 combinations with GA, which demonstrates its great versatility. This indicates that the LOR2 can be an interesting numerical tool to be combined with other GOAs also, to solve challenging optimization tasks.

5. Experiment II: Structural optimization

In order to demonstrate another application of the LOR2 algorithm, a structural design optimization example is presented. Spacecrafts that are required to land without impact on a moon or a planet with gravity, need to have a set of stabilizing legs to maintain the vehicle vertically oriented and facilitate lift-off. These stabilizing legs are structures that consist of a curved cantilever beam, which are required to have low weight, in order to avoid fuel consumption, and are also required to have limited deformation under loading. Achieving a near optimum design configuration for this structure can reduce the elevated costs associated with moon exploration or interplanetary missions, and is an interesting optimization problem. This optimization problem is here used to demonstrate the capacity of the LOR2 algorithm in achieving as an optimization outcome a multiple set of efficient design configurations, providing both efficiency and diversity for SOO.

In order to derive the FEM formulation for the structure, we calculate the energy balance of the system, expressed in terms of displacement functions, according to Ugural [47]. This is given by the variation of elastic energy subtracted by the work performed by field and surface forces.

$$\Delta E = \Delta E_{elastic} - \Delta W_{field} - \Delta W_{surface} = 0 \qquad \text{(eq.10)}$$

If the entire structure is divided in n elements, the elastic potential energy increased on each element is given by the integration on the volume V on each element of the generated strain deformation $\Delta\varepsilon$ multiplied by its corresponding tension σ. The variation of elastic potential energy for the entire system is given,

$$\Delta E_{elastic} = \sum_{1}^{n} \int_{V} (\sigma_x \Delta\varepsilon_x + \sigma_y \Delta\varepsilon_y + \sigma_z \Delta\varepsilon_z) dV \qquad \text{(eq.11)}$$

Similarly, the amount of work performed by field forces and surface forces on the structure can be defined as:

$$\Delta W_{field} = \sum_{1}^{n} \int_{V} (F_x \Delta u + F_y \Delta v + F_z \Delta w)\, dV \tag{eq.12}$$

$$\Delta W_{surface} = \sum_{1}^{n} \int_{s} (p_x \Delta u + p_y \Delta v + p_z \Delta w)\, ds \tag{eq.13}$$

Thus,

$$\Delta E = \sum_{1}^{n} \int_{V} (\sigma_x \Delta\varepsilon_x + \sigma_y \Delta\varepsilon_y + \sigma_z \Delta\varepsilon_z)\, dV - \sum_{1}^{n} \int_{V} (F_x \Delta u + F_y \Delta v + F_z \Delta w)\, dV - \sum_{1}^{n} \int_{s} (p_x \Delta u + p_y \Delta v + p_z \Delta w)\, ds = 0 \tag{eq.14}$$

Using a simplified notation for each element we define:

$$\sum_{1}^{n} \int_{V} (\{\Delta\boldsymbol{\varepsilon}\}_e^T \{\boldsymbol{\sigma}\}_e - \{\Delta \boldsymbol{f}\}_e^T \{\boldsymbol{F}\}_e)\, dV - \sum_{1}^{n} \int_{s} \{\Delta \boldsymbol{f}\}_e^T \{\boldsymbol{p}\}_e ds = 0 \tag{eq.15}$$

If we call the resultant of external forces applied on the nodes on each element of the FEM model $\{\boldsymbol{Q}\}_e$, and substitute $\boldsymbol{\sigma}$ for its expression in terms of nodal displacements, it is defined:

$$\sum_{1}^{n} \{\Delta\boldsymbol{\delta}\}_e^T ([\boldsymbol{k}]_e \{\boldsymbol{\delta}\}_e - \{\boldsymbol{Q}\}_e) = 0 \tag{eq.16}$$

Considering the equilibrium equation can also be defined for one single element, we have,

$$[\boldsymbol{k}]_e \{\boldsymbol{\delta}\}_e - \{\boldsymbol{Q}\}_e = 0 \tag{eq.17}$$

For the entire system, the assembled form of the previous equation is,

$$(\Delta\boldsymbol{\delta})^T ([\boldsymbol{K}]\{\boldsymbol{\delta}\} - \{\boldsymbol{Q}\}) = 0 \tag{eq.18}$$

Since this equation must be satisfied for any arbitrary variations of nodal displacements $(\Delta\boldsymbol{\delta})^T$, it is possible to conclude that,

$$[\boldsymbol{K}]\{\boldsymbol{\delta}\} = \{\boldsymbol{Q}\} \tag{eq.19}$$

Where the global stiffness matrix $[\boldsymbol{K}]$ and the total global nodal force matrix $[\boldsymbol{Q}]$ can be defined as,

$$[\boldsymbol{K}] = \sum_{1}^{n} [\boldsymbol{k}]_e \tag{eq.20}$$

$$[\boldsymbol{Q}] = \sum_{1}^{n} [\boldsymbol{Q}]_e \tag{eq.21}$$

The stabilizing leg is a beam is composed of three sections made of an "I" profile with a hole in the center, where the design configuration of each section can vary under certain limits. A side view of the cantilever beam with the default design configuration, where all design variables are equal to 0.5, is presented in figure 6. The double line at the top and the bottom of each section of the structure are related to the position of the flange of the "I" beam, and the distance between each parallel double line represents the thickness of the flange, which can also vary. The structure parts related to the connection with the spacecraft and the foot of the stabilizing leg are not considered in the FEM analysis, therefore only the central section of the structure is modeled. The total height of the structure is about 1 meter, and it must have limited deformations over its 3 meters length.

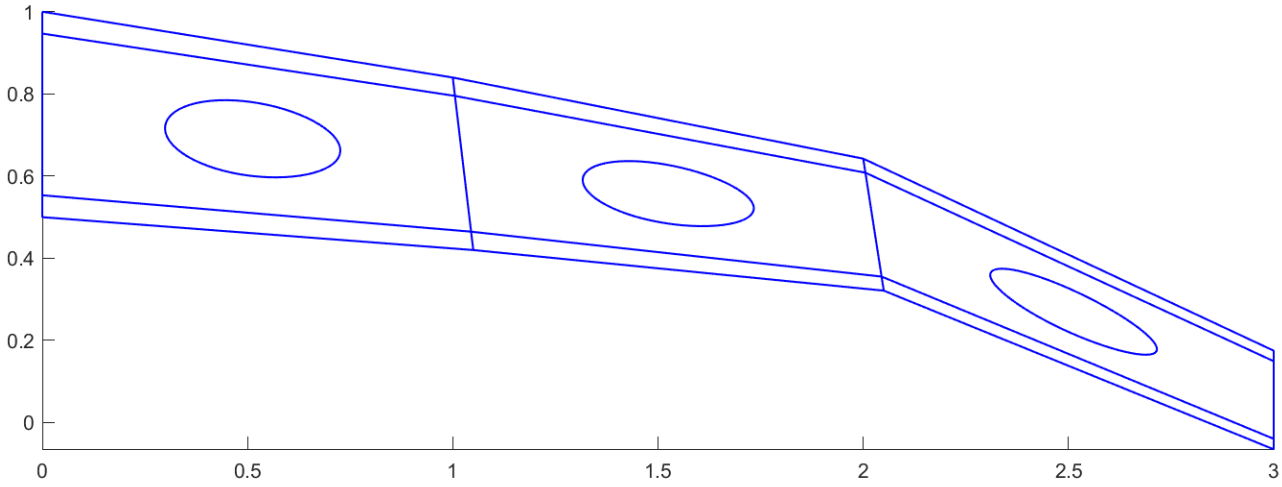

Fig. 6 – The 2D profile of the beam with default configuration (units in m)

In order to calculate the deformations and tensions of the beam, a 2D FEM model with triangular element mesh was generated in MATLAB (figure 7). The optimization algorithm, the mesh generation code, and the FEM processing core were independently coded in MATLAB, thus no MATLAB built-in applications we applied to any of these design steps. The material coefficients used for the model are from the aluminum alloy 2024-T4 [57].

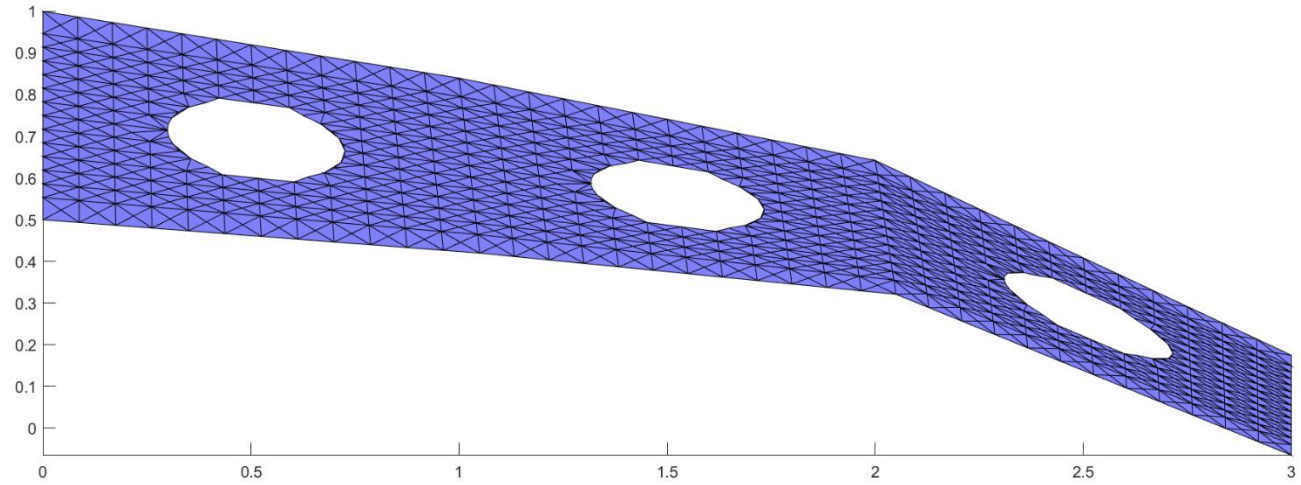

Fig. 7 – The 2D FEM mesh of the beam with default configuration (units in m)

As boundary conditions, the beam is fixed on left side, and a vertical load is placed on the right side of the structure. Under these conditions, the structure must have all nodal displacements limited to a maximum allowed value of 0.05 m. Along all the structure it is necessary to maintain maximum Von Misses principal tension σ_1 lower than the maximum tension allowed, 300 MPa.

For this optimization task, the L2GA-L2PS algorithm, a combination of GA with LOR2, is selected. The goal of the optimization is to generate a diverse set of very efficient design configurations which minimizes the overall structural weight, while maintaining deformations and maximum tensions under defined values. At figure 1 and 2, previously presented, it is seen how the LOR2 algorithm can explore in parallel multiple local optima of a given function. Thus, similarly, the LOR2 algorithm is in this experiment applied for structural optimization, to generate a set of multiple local optima design configurations for the beam, providing both efficiency and diversity for a SOO.

The number of generations of the optimization is set to 800. Part of the operational parameters of the algorithms are presented in table 6, and additional details are provided in Appendix A. Using a computer with quad-core processor AMD A8-7410 APU with 2.2GHz, this optimization task took approximately two days to be completed. Since the algorithm is multi-focus, the intensive localized exploration one each local region is reduced when compared to a regular SMO task, thus it requires more computation time.

Number of generations	:	800
Population size	:	200
Maximum number of apices	:	4
Local optimum region radius (d1)	:	0.100
Redundancy radius (d2)	:	0.050
Allowed replicates per point	:	2

Table 6– Operational parameters for experiment II

With a population size of 200 solutions, 4 allowed apices and 2 allowed replicates, it is reduced the number of solutions near the apices of each local optimum. This generates less conglomeration and an increased probability of exploring multiple regions inside each local optimum, to reach the best local optima. The downside is that it reduces the intensive localized exploration near each apex, thus requiring more iterations for the optimization to converge.

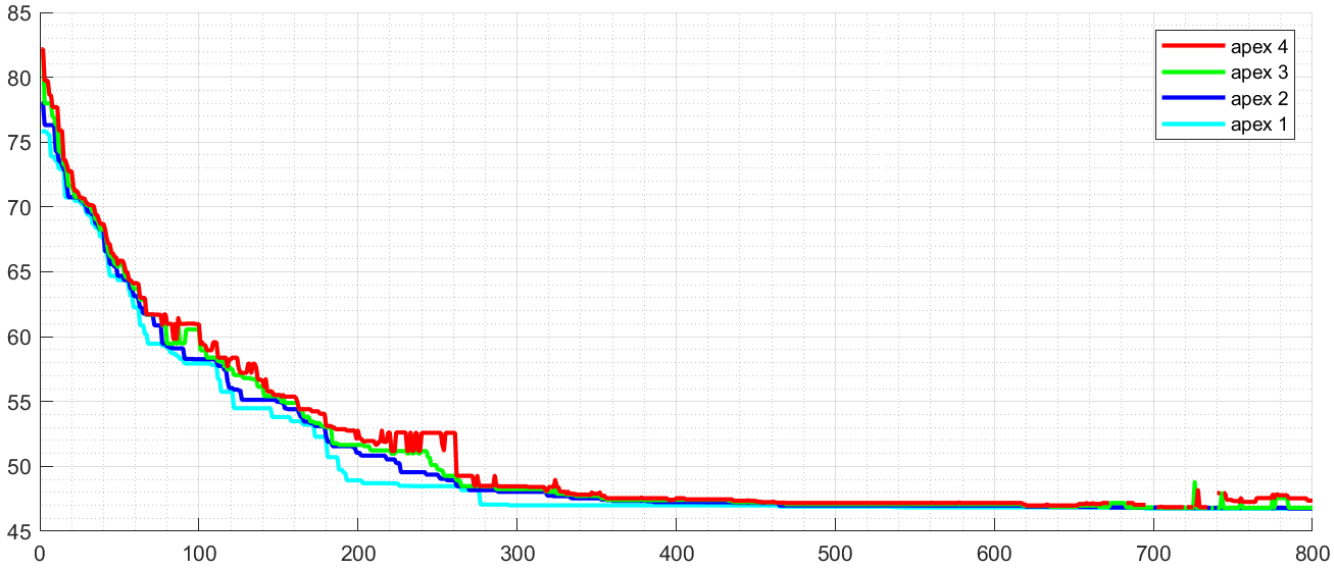

Fig. 8 – Convergion curve

Figure 8 displays the convergion curve of the 4 apices along the 800 generations. It is possible to see that the primary apex in light blue has a good convergion without fluctuation, while apices 2, 3, and 4 present some fluctuation along the generations. This is because when an apex optimizes, it dislocates its position inside the design space, and when it comes closer to an existing apex with lower objective value, this lower leader becomes a sub solution of the leader with better objective value. Thus, some apices can be suppressed along the optimization, and other further solutions with lower objective value are considered other local apices. Since some apices can be suppressed and not have other solutions to replace them, it is interesting to record the last population of solutions along the generations which presented all the allowed apices, to assure the desired diversity and efficiency output for the optimization task.

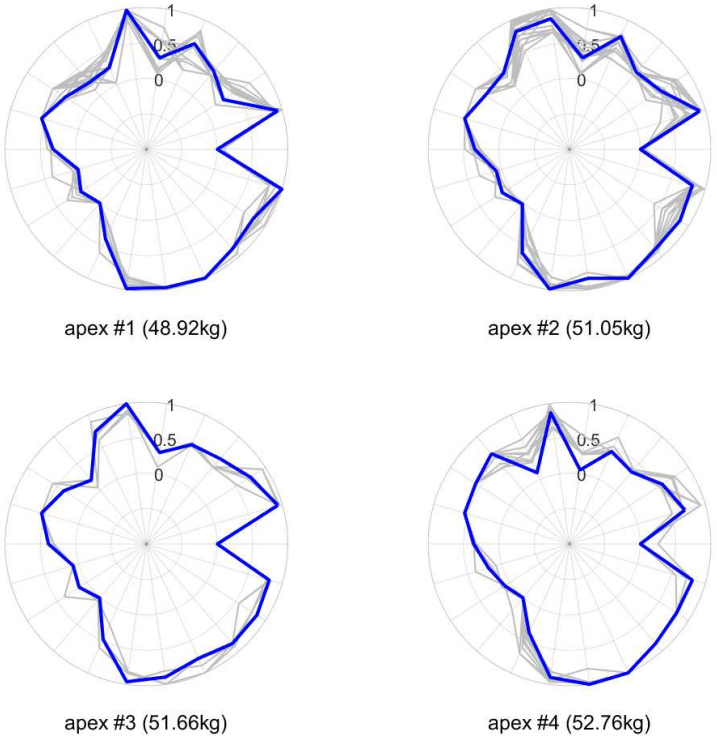

Fig. 9 – Optimized design polygons - Apices 1 to 4 (in blue) and sub solutions (in gray)

Figure 9 displays the design polygons of the achieved design configurations. The apices of the 4 local regions achieved represent 4 main design configurations, which are presented in blue, while the sub solutions of each 4 local optima are presented in gray. The best design solution achieved is leader 1 with 48.92 kg as objective value. Very close to this value are leader 2, 3, and 4 with 51.05 kg, 51.66 kg, and 52.76 kg respectively. The outline of the achieved design configurations in this optimization study are displayed in figure 10.

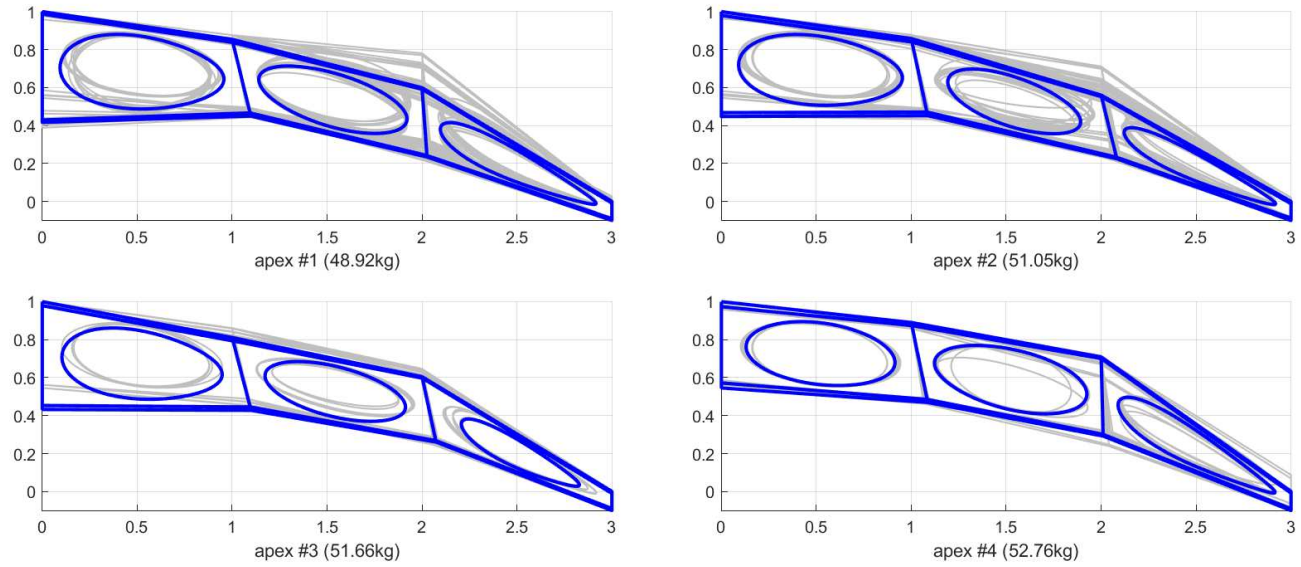

Fig. 10 – Optimized design configurations (m) - Apices 1 to 4 (in blue) and sub solutions (in gray)

These solutions displayed in blue are the best local optima configurations, the apices of each local optima, which are the 4 most competitive solutions in terms of weight efficiency while having significant distance between each other in the design space. This distance between each leader is of at least d_1 in the normalized design space, which translates into different design configurations for the structure. This illustrates that a single optimization study using a GOA combined with LOR2 algorithm, with a combination that has segregated populations, can generate distinct local apices designs with competitive performance, and several sub solutions. This multi-focus approach provides diversified solutions and also reduces the probability of the optimization process getting trapped in a poor local optimum, as can happen in the single modal optimization. With these mentioned advantages, the LOR2 algorithm is expected to give important contributions to several industrial process of advanced design.

Figure 11 displays the field distribution of principal stress σ_1 measured in Pascals (Pa). It is possible to see that all maximum stresses are less than 300 MPa, as a design requirement. As expected, the greater concentration of stress occurs near thin regions of the structure.

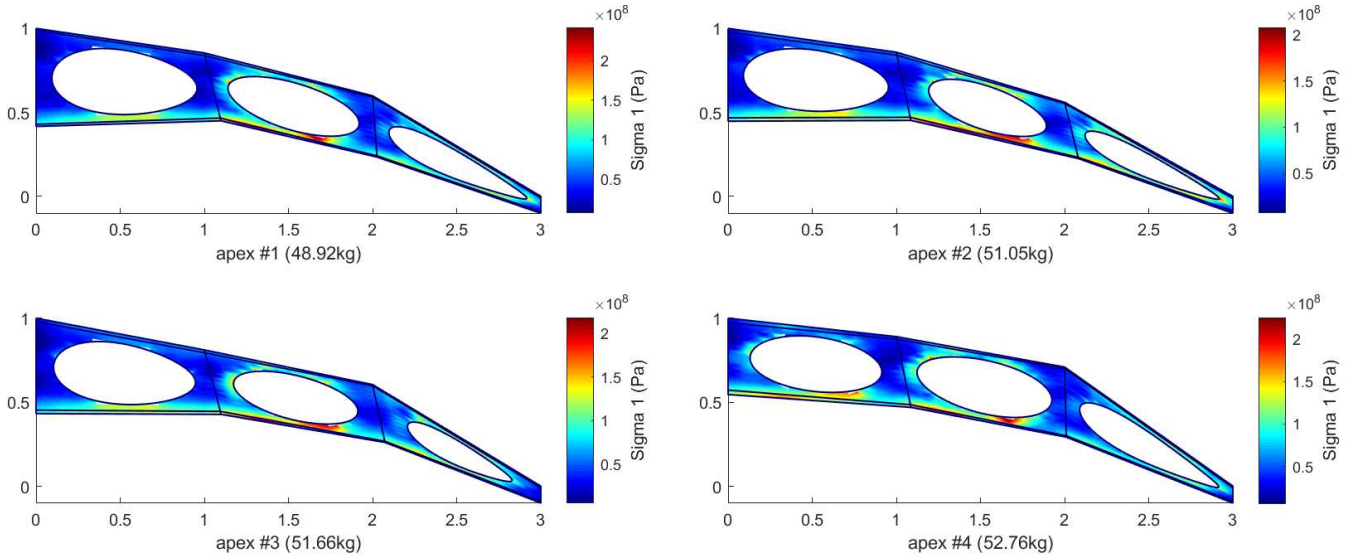

Fig. 11 – Optimized design configurations (m), principal stresses σ_1 (Pa) - Apices 1 to 4

Figure 12 displays the field distribution of total displacement under loading in meters. It is possible to note that the maximum displacements for all design configurations are less than 0.05 m, as the second design requirement.

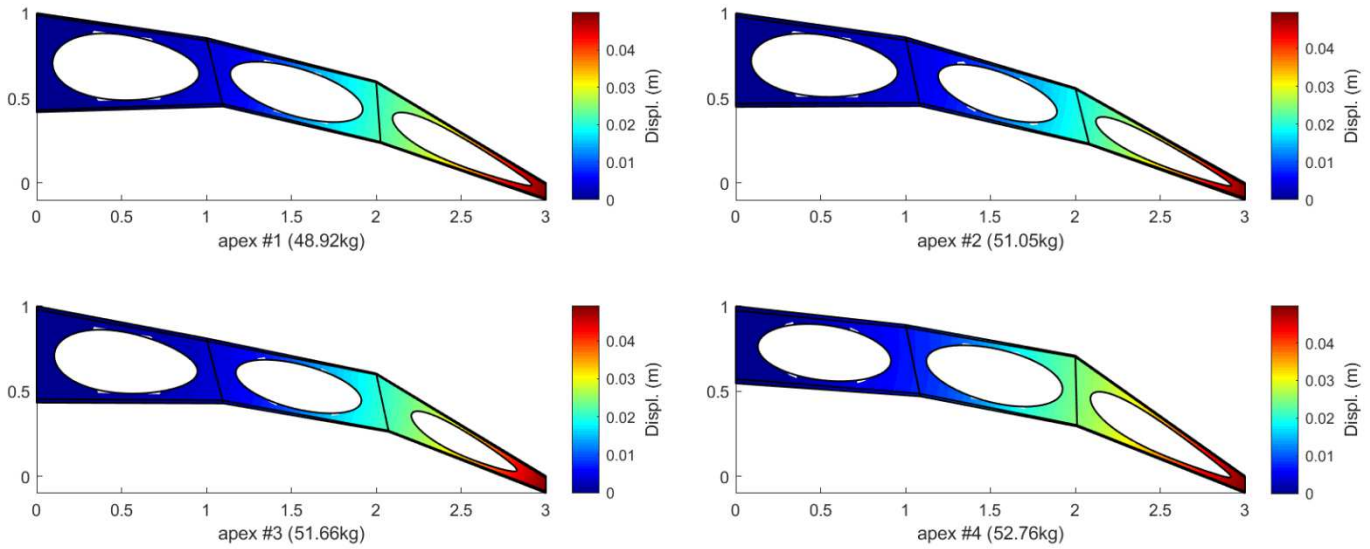

Fig. 12 – Optimized design configurations (m) – Displacement field (m) - Apices 1 to 4

The optimization of a cantilever beam under loading is performed with the L2GA-L2PS algorithm, a combination of the GA and LOR2, having a GA with segregated population - which means the cross-over process occurs only between solutions of the same local optimum region. As expected, the optimization task provided 4 different efficient designs as output, due to its multi-focus approach. With this multi-focus approach, it is generated a set of efficient configurations, which provides options for the design process, and a lower probability of having the optimization process getting trapped in poor local optima. The graphical analysis of structure profiles demonstrates the designs indeed differ, having several sub solutions, and the plots of σ_1 stresses and nodal displacements confirming the designs are under the specified requirements.

This optimization experiment showcases the advantages of the multi-focus approach of the LOR2 algorithm when compared to single-focus methods. As the results demonstrate, it is expected that the LOR2 algorithm can provide important contributions to several design challenges of SOO.

6. Experiment III: Metamodel segmentation

Another application of the LOR2 algorithm is to segment response surfaces. In order to compare the efficiency of segmented and non-segmented response surfaces, several benchmark functions used for this purpose are selected [2]. The metamodel applied in this experiment is the model called Kriging (KR), also called Gauss Process, which is a popular metamodel in AI and optimization used to forecast the output of a function, well known for being very precise, yet also very expensive in terms of computation costs for large datasets. Each function has its 2D domain defined in a Cartesian grid of 71 x 71 points, and the error of the output and other performance parameters are presented. Tables 7 to 10 display the comparison in terms of performance of the metamodel and time delayed on training. The metrics for metamodel performance measurement are the average error and variance of the error. Other metrics to measure the performance of metamodels are also applied, which are R square, RMAE, and RAAE, as described by Mahdi et al. [20]. It is important to note that for the response surface mesh (RSM), each element of the metamodel was training in parallel computing using a dual-core computer, and the overall training time includes the mesh generation. The kriging with response surface mesh approach (RSM-KR) has its performance compared with the regular KR application for four functions:

Ackley function		
response surface	KR	RSM-KR
n segments	1	130
total training time	7min 14.8sec	10.98sec
mean_error	4.537e-01	1.397e-03
variance	3.258e-01	8.733e-05
r square	9.968e-01	1.000e+00
raae	4.502e-02	1.385e-04
rmae	2.737e-01	4.036e-02

Table 7 – Performance comparison for the Ackley function

Beale function		
response surface	KR	RSM-KR
n segments	1	222
total training time	20min 3.88sec	12.62sec
mean_error	2.080e-01	1.732e-01
variance	9.528e-02	1.765e-01
r square	1.000e+00	1.000e+00
raae	1.615e-04	1.345e-04
rmae	3.026e-03	3.710e-03

Table 8 – Performance comparison for the Beale function

Booth function		
response surface	KR	RSM-KR
n segments	1	225
total training time	27min 1.21sec	12.81sec
mean_error	4.827e-02	1.722e-01
variance	3.659e-03	1.746e-01
r square	1.000e+00	1.000e+00
raae	7.777e-05	2.775e-04
rmae	3.538e-04	4.543e-03

Table 9 – Performance comparison for the Booth function

Custom probability density function		
response surface	KR	RSM-KR
n segments	1	209
total training time	15min 6.51sec	12.47sec
mean_error	1.640e-04	1.913e-04
variance	1.476e-07	9.837e-08
r square	1.000e+00	1.000e+00
raae	8.192e-04	9.557e-04
rmae	1.238e-02	1.290e-02

Table 10 – Performance comparison for the Custom Probability Density Function

From table 7 to 10 is possible to see that the training time of the KR model, which ranged from 27 to 7 minutes, was significantly reduced with the generation of the RSM with the LOR2. The RSM-KR presented a total computation time of about 11 to 13 seconds, a time reduction up to more than 120 times. These results were achieved with niche generation with d_1 equal to 0.1, and overlapping PMEs, with overlap radius set as d_1 from the PV. Experiments have demonstrated that for a domain with an increased number of inputs, the training time for the non-segmented model can achieve many hours, while the segmented mesh remains less than 20 seconds for the tested cases. Also, it is noted that there is not much difference in terms of the precision of the RSM, which demonstrates that the LOR2 can be applied to generate a RSM, and is an important alternative to reduce computation costs of metamodels.

7. Concluding Remarks

GOAs presented a great advantage when compared with gradient-based optimization since these methods allow a better exploration of the overall design space. However, most GOAs converge to a single point in the design space, what can bring large variance values for the output, since many functions present several local optima. Thus, many niching methods were created to achieve a diversified set of solutions, and identify the diverse local optima. The LOR2 presents important advantages before existing niching methods, like the identification of the most suitable and interesting solutions among all the population, and an equalized exploration effort on each local optimum region. Also, it has a great versatility in giving many ways to be combined with GOAs, since it provides several metrics of the population of solutions, which generates very efficient alternative

methods. The LOR2 has proven in this study to outperform state-of-the-art methodologies for the greater number of test functions, as described in the first experiment.

It is also important to note that the experiments here described considered only the application of the LOR2 with the parallel exploration of selected local optima. Yet for many functions is possible to use the LOR2 to discretize the domain by listing and locating each local optimum that presented the most competitive responses. This can facilitate the methodical thorough exploration of the function, and provide results with increased efficiency than those presented in this study.

The LOR2 demonstrated its capacity of achieving distinct and efficient design configuration in the multimodal design optimization of a metallic cantilever beam. Since the four main solution are identified, it signifies an important advancement in MMO, by avoiding the need to verify manually a large number of possible solutions as the optimization output.

Metamodels such as KR, which is very popular for its precision yet very expensive, are significantly simplified and have their computation cost greatly reduced when partitioned. As the experiment demonstrate, the LOR2 is a good alternative to segment a dataset into regions of similitude. This LOR2 metamodel partitioning can be applied to any metamodel, and response surface elements can be trained or updated independently, and datasets size and precision can be significantly increased. Thus the LOR2 provides increased operational speed and efficiency to all AI and metamodel aided optimization tasks. It is interesting to note that the LOR2 RSM generation was coded in MATLAB for these experiments, and the mesh generation was responsible for 95% of the computation time of the RSM. Yet it can perform significantly faster if coded in a faster computation language such C++ or JAVA.

Given the versatility and efficiency of the LOR2, it is expected that the LOR2 will benefit several advanced design tasks and optimization challenges in general, especially for functions with large domains that present multiple global optima. Also, it can help in reducing computation costs of AI tasks in general, significantly improving the performance of many AI tasks.

References

[1] R. Jin, W. Chen, and T. W. Simpson, "Comparative studies of metamodeling techniques under multiple modeling criteria," *8th Symp. Multidiscip. Anal. Optim.*, 2000, doi: 10.2514/6.2000-4801.

[2] C. Bogoclu and D. Roos, "A benchmark of contemporary metamodeling algorithms," *ECCOMAS Congr. 2016 - Proc. 7th Eur. Congr. Comput. Methods Appl. Sci. Eng.*, vol. 2, no. June, pp. 3344–3360, 2016, doi: 10.7712/100016.2039.7645.

[3] D. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning," *Addison-Wesley Prof.*, 1989.

[4] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proc. IEEE Int. Conf. Neural Networks*, pp. 1942–1948, 1995.

[5] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, 1997, doi: 10.1023/A:1008202821328.

[6] T. H. A. Scollen, "Simulated Annleaing, Introduction, Application and Theory," *Nove Sci. Publ. Inc. New York*, 2018.

[7] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems," *Eng. Comput.*, vol. 29, no. 1, pp. 17–35, 2013, doi: 10.1007/s00366-011-0241-y.

[8] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. of Michigan Press, 1975.

[9] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," *Proc. ofthe Second Int. Conf. Genet. Algorithms*, pp. 41–49, 1987.

[10] G. R. Harik, "Finding multimodal solutions using restricted tournament selection," *Proc. Sixth Int. Conf. Genet. Algorithms*, 1995, [Online]. Available: citeseer.ist.psu.edu/.

[11] S. W. Mahfoud, "Crowding and preselection revisited," *Parallel Probl. solving from Nat. 2*, 1992, [Online]. Available: citeseer.ist.psu.edu/mahfoud92crowding.html.

[12] K. A. De Jong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems.," University of Michigan, USA, 1975.

[13] D. Beasley, D. R. Bull, and R. R. Martin, "A sequential niche technique for multimodal function optimization," *Evol. Comput.*, 1993, [Online]. Available: citeseer.ist.psu.edu/beasley93sequential.html.

[14] M. Bessaou, A. Petrowski, and P. Siarry, "Island Model Cooperating with Speciation for Multimodal Optimization," *Springer Berlin Heidelb.*, 2000.

[15] X. Yin and N. Germay, "A fast genetic algorithm with sharing scheme using cluster analysis methods in multi-modal function optimization," *Int. Conf. Artif. Neural Networks Genet. Algorithms*, pp. 450–457, 1993.

[16] "World Congress of Computational Intelligence." 2020, [Online]. Available: https://wcci2020.org.

[17] "Congress of Evolutionary Computation - Niching Competition." 2020, [Online]. Available: http://epitropakis.co.uk/cec20-niching/competition.

[18] Marde Helbig,´ A. P. E., and M. Optimization, "Benchmark Functions for CEC 2015 Special Session and Competition on Dynamic Multi-objective Optimization," pp. 1–7, 2015.

[19] X. Li, M. G. Epitropakis, K. Deb, and A. Engelbrecht, "Seeking Multiple Solutions: An Updated Survey on Niching Methods and Their Applications," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 518–538, 2017, doi: 10.1109/TEVC.2016.2638437.

[20] B. Sareni and L. Krahenbuhl, "Fitness sharing and niching methods revisited," *IEEE Trans. Evol. Comput.*, vol. 2, no. 3, pp. 97–106, 1998, doi: 10.1109/4235.735432.

[21] A. Passaro and A. Starita, "Niching in Particle Swarm Optimization," *Dep. Comput. Sci.*, vol. Doctor of, no. June, p. 133, 2007.

[22] K.-C. Wong, "Evolutionary Multimodal Optimization: A Short Survey," *ArXiv*, vol. abs/1508.0, 2015.

[23] C. Yue *et al.*, "Differential evolution using improved crowding distance for multimodal multiobjective optimization," *Swarm Evol. Comput.*, vol. 62, p. 100849, 2021, doi: https://doi.org/10.1016/j.swevo.2021.100849.

[24] J. Liang *et al.*, "A clustering-based differential evolution algorithm for solving multimodal multi-objective optimization problems," *Swarm Evol. Comput.*, vol. 60, p. 100788, 2021, doi: https://doi.org/10.1016/j.swevo.2020.100788.

[25] Y. Wu, W. Ma, Q. Miao, and S. Wang, "Multimodal continuous ant colony optimization for multisensor remote sensing image registration with local search," *Swarm Evol. Comput.*, vol. 47, pp. 89–95, 2019, doi: https://doi.org/10.1016/j.swevo.2017.07.004.

[26] R. Poláková, J. Tvrdík, and P. Bujok, "Differential evolution with adaptive mechanism of population size according to current population diversity," *Swarm Evol. Comput.*, vol. 50, p. 100519, 2019, doi: https://doi.org/10.1016/j.swevo.2019.03.014.

[27] K. Deb and S. Gulati, "Design of truss-structures for minimum weight using genetic algorithms," *Finite Elem. Anal. Des.*, vol. 37, no. 5, pp. 447–465, 2001, doi: https://doi.org/10.1016/S0168-874X(00)00057-3.

[28] K. Deb, "Optimal Design of a Class of Welded Structures via Genetic Algorithms," in *31st Structures, Structural Dynamics and Materials Conference*, 1990.

[29] W. J. de S. Gomes and A. T. Beck, "Global structural optimization considering expected consequences of failure and using ANN surrogates," *Comput. Struct.*, vol. 126, pp. 56–68, 2013, doi: https://doi.org/10.1016/j.compstruc.2012.10.013.

[30] D. Wang, W. H. Zhang, and J. S. Jiang, "Truss Optimization on Shape and Sizing with Frequency Constraints," *AIAA J.*, vol. 42, no. 3, pp. 622–630, 2004, doi: 10.2514/1.1711.

[31] A. Kaveh and S. Talatahari, "Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss

structures," *Comput. Struct.*, vol. 87, no. 5, pp. 267–283, 2009, doi: https://doi.org/10.1016/j.compstruc.2009.01.003.
[32] A. Kaveh and M. Ilchi Ghazaan, "Hybridized optimization algorithms for design of trusses with multiple natural frequency constraints," *Adv. Eng. Softw.*, vol. 79, pp. 137–147, 2015, doi: https://doi.org/10.1016/j.advengsoft.2014.10.001.
[33] A. Kaveh and A. Zolghadr, "Truss optimization with natural frequency constraints using a hybridized CSS-BBBC algorithm with trap recognition capability," *Comput. Struct.*, vol. 102–103, pp. 14–27, 2012, doi: https://doi.org/10.1016/j.compstruc.2012.03.016.
[34] A. Kaveh and S. Talatahari, "Size optimization of space trusses using Big Bang–Big Crunch algorithm," *Comput. Struct.*, vol. 87, no. 17, pp. 1129–1140, 2009, doi: https://doi.org/10.1016/j.compstruc.2009.04.011.
[35] S. K. Azad and O. Hasançebi, "An elitist self-adaptive step-size search for structural design optimization," *Appl. Soft Comput.*, vol. 19, pp. 226–235, 2014, doi: https://doi.org/10.1016/j.asoc.2014.02.017.
[36] W. Lingyun, Z. Mei, W. Guangming, and M. Guang, "Truss optimization on shape and sizing with frequency constraints based on genetic algorithm," *Comput. Mech.*, vol. 35, no. 5, pp. 361–368, 2005, doi: 10.1007/s00466-004-0623-8.
[37] H. M. Gomes, "Truss optimization with dynamic constraints using a particle swarm algorithm," *Expert Syst. Appl.*, vol. 38, no. 1, pp. 957–968, 2011, doi: https://doi.org/10.1016/j.eswa.2010.07.086.
[38] R. E. Perez and K. Behdinan, "Particle swarm approach for structural design optimization," *Comput. Struct.*, vol. 85, no. 19, pp. 1579–1588, 2007, doi: https://doi.org/10.1016/j.compstruc.2006.10.013.
[39] J. Cheng, "Optimum design of steel truss arch bridges using a hybrid genetic algorithm," *J. Constr. Steel Res.*, vol. 66, no. 8, pp. 1011–1017, 2010, doi: https://doi.org/10.1016/j.jcsr.2010.03.007.
[40] C.-Y. Lin and W.-T. Chen, "Stochastic multistage algorithms for multimodal structural optimization," *Comput. Struct.*, vol. 74, no. 2, pp. 233–241, 2000, doi: https://doi.org/10.1016/S0045-7949(99)00016-4.
[41] M. J. Islam, X. Li, and K. Deb, "Multimodal Truss Structure Design Using Bilevel and Niching Based Evolutionary Algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017, pp. 274–281, doi: 10.1145/3071178.3071251.
[42] L. Huang, C.-T. Ng, A. H. Sheikh, and M. C. Griffith, "Niching particle swarm optimization techniques for multimodal buckling maximization of composite laminates," *Appl. Soft Comput.*, vol. 57, pp. 495–503, 2017, doi: https://doi.org/10.1016/j.asoc.2017.04.006.
[43] K. Martini, "Harmony Search Method for Multimodal Size, Shape, and Topology Optimization of Structural Frameworks," *J. Struct. Eng.*, 2011.
[44] J. A. Czyz and S. A. Lukasiewicz, "Multimodal optimization of structures with frequency constraints," doi: https://doi.org/10.2514/3.12573.
[45] C.-Y. Lin and I.-M. Huang, "A topographical approach for multimodal structural optimal designs," *Trans. Built Environ.*, vol. 28, 1997.
[46] R. D. Cook, D. S. Malkus, and M. E. Plesha, *Concepts and Applications of Finite Element Analysis*, Third Edit. New York: John Wiley & Sons, 1988.
[47] A. C. Ugural and S. K. Fenster, *Advanced Mechanics of Materials and Applied Elasticity*, Fifth edit. Prentice Hall, 2012.
[48] K. Liu and A. Tovar, "An efficient 3D topology optimization code written in Matlab," *Struct. Multidiscip. Optim.*, vol. 50, no. 6, pp. 1175–1196, 2014, doi: 10.1007/s00158-014-1107-x.
[49] A. Gauchia, V. Diaz, M. J. L. Boada, and B. L. Boada, "Torsional stiffness and weight optimization of a real bus structure," *Int. J. Automot. Technol.*, vol. 11, no. 1, pp. 41–47, 2010, doi: 10.1007/s12239-010-0006-4.
[50] H.-S. Park and X.-P. Dang, "Structural optimization based on CAD–CAE integration and metamodeling techniques," *Comput. Des.*, vol. 42, no. 10, pp. 889–902, 2010, doi: https://doi.org/10.1016/j.cad.2010.06.003.
[51] G. Zheng, H. Nie, J. Chen, C. Chen, and H. P. Lee, "Dynamic analysis of lunar lander during soft landing using explicit finite element method," *Acta Astronaut.*, vol. 148, pp. 69–81, 2018, doi: https://doi.org/10.1016/j.actaastro.2018.04.014.
[52] C. E. Rasmussen, "Gaussian Processes in Machine Learning," in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, O. Bousquet, U. von Luxburg, and G. Rätsch, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 63–71.
[53] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986, doi: 10.1038/323533a0.
[54] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553. Nature Publishing Group, pp. 436–444, May 2015, doi: 10.1038/nature14539.
[55] J. N. Reddy, "An Introduction to the Finite Element Method," *J. Press. Vessel Technol.*, vol. 111, no. 3, pp. 348–349, 1989, doi: 10.1115/1.3265687.
[56] A. Jara, "Theory and computations for the Dirichlet process and related models: An overview," *Int. J. Approx. Reason.*, vol. 81, pp. 128–146, 2017, doi: https://doi.org/10.1016/j.ijar.2016.11.008.
[57] "ASM Aerospace Specifications Metals Inc. - Aluminum Alloy 2024-T4," 2000. http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=MA2024T4.

Appendix A – Operational parameters of GOAs

The GA and PSO algorithms have the following parameters for their operation, in all its combinations.

- **GA**

GA Parameters	value	comment
Ratio of mutated pop	0.5	between 0 and 1
Mutation probability	0.02	between 0 and 1
Max mutation range	0.25	between 0 and 1

Table A1– GA parameters

The GA operates with the random pair selection. In the L2GA and L2GA-L2PS variations, the pair selection is still random, yet it occurs only inside the same local optimum. Experiments have demonstrated that generating individuals either mutated or recombined provide increased efficiency than generating individual solutions both mutated and recombined. In this study, the selected ratio of the mutated population size from the overall generated population is 0.5, as described in table A1.

- **PSO**

PSO Parameters	value	comment
global best speed	2	
local best speed	2	
Maximum speed	0.5	between 0 and 1
Initial weight	1	

Table A2– PSO parameters

The PSO parameters are presented in table A2. The mass weight of each particle in the PSO algorithm is reduced along the generations, by being multiplied by the coefficient iwt. This coefficient is also adapted to different values of $iter_{max}$, according to equation A.1:

$$iwt = 0.9 - \frac{iter}{2\ iter_{max}} \tag{A.1}$$

Appendix B – Optimization results

The detailed optimization results are presented, to the five accuracy values, from 1.0 E-01 to 1.0 E-05. For each table, PR and SR are defined over 50 optimization runs, following the equations 6 and 7.

Results per accuracy = 1E-01

function	F1(1D)		F2(1D)		F3(1D)		F4(2D)		F5(2D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	1.000	1.000	1.000	1.000	1.000	1.000	0.640	0.300	1.000	1.000
L2GA-L2PS	1.000	1.000	1.000	1.000	1.000	1.000	0.695	0.380	1.000	1.000
L2GA	0.620	0.240	0.936	0.920	1.000	1.000	0.255	0.000	0.970	0.940
L2PSO	0.960	0.940	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
GA-C	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
GA-DC	0.700	0.400	0.904	0.880	1.000	1.000	0.315	0.080	1.000	1.000
GA-RTS	1.000	1.000	0.992	0.980	1.000	1.000	0.885	0.720	1.000	1.000

function	F6(2D)		F7(2D)		F6(3D)		F7(3D)		F8(2D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	0.148	0.000	0.989	0.960	0.001	0.000	0.358	0.000	1.000	1.000
L2GA-L2PS	0.026	0.000	0.867	0.340	0.000	0.000	0.369	0.000	0.998	0.980
L2GA	0.047	0.000	1.000	1.000	0.004	0.000	0.881	0.880	0.085	0.000
L2PSO	0.448	0.000	1.000	1.000	0.315	0.000	0.217	0.000	1.000	1.000
GA-C	0.138	0.000	1.000	1.000	0.007	0.000	1.000	1.000	1.000	1.000
GA-DC	0.047	0.000	1.000	1.000	0.006	0.000	1.000	1.000	0.138	0.060
GA-RTS	0.076	0.000	0.982	0.980	0.005	0.000	0.473	0.460	0.310	0.060

function	F9(2D)		F10(2D)		F11(2D)		F11(3D)		F12(3D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	0.240	0.000	0.018	0.000	0.120	0.000	0.177	0.000	0.095	0.000
L2GA-L2PS	0.243	0.000	0.030	0.000	0.067	0.000	0.210	0.000	0.075	0.000
L2GA	0.167	0.000	0.113	0.000	0.167	0.000	0.167	0.000	0.105	0.000
L2PSO	1.000	1.000	0.600	0.000	0.673	0.240	1.000	1.000	1.000	1.000
GA-C	0.603	0.100	0.233	0.000	0.327	0.000	0.950	0.920	0.980	0.980
GA-DC	0.170	0.000	0.125	0.000	0.167	0.000	0.167	0.000	0.122	0.000
GA-RTS	0.350	0.000	0.145	0.000	0.250	0.000	0.357	0.000	0.245	0.000

function	F11(5D)		F12(5D)		F11(10D)		F12(10D)		F12(20D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	0.167	0.000	0.020	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2GA-L2PS	0.193	0.000	0.040	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2GA	0.160	0.000	0.045	0.000	0.167	0.000	0.005	0.000	0.003	0.000
L2PSO	1.000	1.000	1.000	1.000	1.000	1.000	0.618	0.000	0.000	0.000
GA-C	1.000	1.000	0.163	0.160	1.000	1.000	0.003	0.000	0.033	0.000
GA-DC	0.163	0.000	0.035	0.000	0.233	0.080	0.010	0.000	0.005	0.000
GA-RTS	0.420	0.100	0.250	0.000	0.837	0.640	0.270	0.000	0.000	0.000

Results per accuracy = 1E-02

function	F1(1D)		F2(1D)		F3(1D)		F4(2D)		F5(2D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	1.000	1.000	0.992	0.960	0.860	0.860	0.100	0.000	1.000	1.000
L2GA-L2PS	1.000	1.000	0.984	0.920	0.860	0.860	0.095	0.000	0.990	0.980
L2GA	0.620	0.240	0.232	0.040	1.000	1.000	0.205	0.000	0.630	0.260
L2PSO	0.960	0.940	0.988	0.980	1.000	1.000	1.000	1.000	1.000	1.000
GA-C	1.000	1.000	0.976	0.880	0.860	0.860	0.595	0.320	1.000	1.000
GA-DC	0.680	0.360	0.252	0.060	0.980	0.980	0.215	0.000	0.730	0.460
GA-RTS	1.000	1.000	0.684	0.180	0.960	0.960	0.260	0.000	1.000	1.000

function	F6(2D)		F7(2D)		F6(3D)		F7(3D)		F8(2D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	0.014	0.000	0.736	0.000	0.000	0.000	0.331	0.000	1.000	1.000
L2GA-L2PS	0.001	0.000	0.671	0.000	0.000	0.000	0.275	0.000	0.810	0.120
L2GA	0.017	0.000	0.028	0.000	0.000	0.000	0.005	0.000	0.085	0.000
L2PSO	0.426	0.000	0.436	0.000	0.298	0.000	0.209	0.000	1.000	1.000
GA-C	0.018	0.000	0.984	0.860	0.000	0.000	0.997	0.980	0.995	0.960
GA-DC	0.026	0.000	0.028	0.000	0.000	0.000	0.005	0.000	0.083	0.000
GA-RTS	0.028	0.000	0.108	0.000	0.000	0.000	0.026	0.000	0.238	0.000

function	F9(2D)		F10(2D)		F11(2D)		F11(3D)		F12(3D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	0.023	0.000	0.003	0.000	0.003	0.000	0.010	0.000	0.007	0.000
L2GA-L2PS	0.030	0.000	0.005	0.000	0.003	0.000	0.013	0.000	0.010	0.000
L2GA	0.157	0.000	0.077	0.000	0.130	0.000	0.160	0.000	0.105	0.000
L2PSO	0.630	0.000	0.560	0.000	0.527	0.000	0.687	0.000	0.682	0.000
GA-C	0.167	0.000	0.068	0.000	0.097	0.000	0.167	0.000	0.125	0.000
GA-DC	0.150	0.000	0.090	0.000	0.137	0.000	0.167	0.000	0.122	0.000
GA-RTS	0.140	0.000	0.058	0.000	0.090	0.000	0.253	0.000	0.135	0.000

function	F11(5D)		F12(5D)		F11(10D)		F12(10D)		F12(20D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	0.007	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2GA-L2PS	0.000	0.000	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2GA	0.157	0.000	0.045	0.000	0.163	0.000	0.000	0.000	0.000	0.000
L2PSO	0.667	0.000	0.495	0.000	0.413	0.000	0.130	0.000	0.000	0.000
GA-C	0.167	0.000	0.043	0.000	0.187	0.000	0.000	0.000	0.000	0.000
GA-DC	0.167	0.000	0.025	0.000	0.167	0.000	0.000	0.000	0.000	0.000
GA-RTS	0.197	0.000	0.125	0.000	0.070	0.000	0.010	0.000	0.000	0.000

Results per accuracy = 1E-03

function	F1(1D)		F2(1D)		F3(1D)		F4(2D)		F5(2D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	1.000	1.000	0.780	0.260	0.380	0.380	0.015	0.000	0.690	0.540
L2GA-L2PS	1.000	1.000	0.744	0.180	0.560	0.560	0.005	0.000	0.770	0.620
L2GA	0.620	0.240	0.200	0.000	0.900	0.900	0.065	0.000	0.500	0.000
L2PSO	0.960	0.940	0.988	0.980	1.000	1.000	1.000	1.000	1.000	1.000
GA-C	1.000	1.000	0.784	0.300	0.460	0.460	0.070	0.000	0.910	0.860
GA-DC	0.670	0.340	0.200	0.000	0.760	0.760	0.105	0.000	0.520	0.040
GA-RTS	1.000	1.000	0.472	0.000	0.620	0.620	0.060	0.000	0.740	0.480

function	F6(2D)		F7(2D)		F6(3D)		F7(3D)		F8(2D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	0.000	0.000	0.586	0.000	0.000	0.000	0.293	0.000	0.717	0.280
L2GA-L2PS	0.001	0.000	0.379	0.000	0.000	0.000	0.131	0.000	0.303	0.000
L2GA	0.003	0.000	0.028	0.000	0.000	0.000	0.005	0.000	0.085	0.000
L2PSO	0.423	0.000	0.403	0.000	0.257	0.000	0.165	0.000	1.000	1.000
GA-C	0.002	0.000	0.719	0.080	0.000	0.000	0.936	0.500	0.737	0.320
GA-DC	0.004	0.000	0.028	0.000	0.000	0.000	0.005	0.000	0.083	0.000
GA-RTS	0.001	0.000	0.107	0.000	0.000	0.000	0.025	0.000	0.200	0.000

function	F9(2D)		F10(2D)		F11(2D)		F11(3D)		F12(3D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2GA-L2PS	0.000	0.000	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2GA	0.073	0.000	0.028	0.000	0.033	0.000	0.097	0.000	0.068	0.000
L2PSO	0.550	0.000	0.515	0.000	0.510	0.000	0.673	0.000	0.677	0.000
GA-C	0.047	0.000	0.018	0.000	0.043	0.000	0.087	0.000	0.095	0.000
GA-DC	0.077	0.000	0.018	0.000	0.067	0.000	0.093	0.000	0.098	0.000
GA-RTS	0.050	0.000	0.005	0.000	0.027	0.000	0.047	0.000	0.048	0.000

function	F11(5D)		F12(5D)		F11(10D)		F12(10D)		F12(20D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2GA-L2PS	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2GA	0.063	0.000	0.022	0.000	0.020	0.000	0.000	0.000	0.000	0.000
L2PSO	0.667	0.000	0.487	0.000	0.230	0.000	0.052	0.000	0.000	0.000
GA-C	0.067	0.000	0.018	0.000	0.013	0.000	0.000	0.000	0.000	0.000
GA-DC	0.070	0.000	0.028	0.000	0.027	0.000	0.000	0.000	0.000	0.000
GA-RTS	0.027	0.000	0.018	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Results per accuracy = 1E-04

function	F1(1D)		F2(1D)		F3(1D)		F4(2D)		F5(2D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	1.000	1.000	0.376	0.040	0.080	0.080	0.000	0.000	0.250	0.040
L2GA-L2PS	1.000	1.000	0.420	0.020	0.260	0.260	0.000	0.000	0.250	0.040
L2GA	0.620	0.240	0.176	0.000	0.480	0.480	0.010	0.000	0.470	0.000
L2PSO	0.940	0.900	0.860	0.760	1.000	1.000	1.000	1.000	1.000	1.000
GA-C	1.000	1.000	0.360	0.020	0.200	0.200	0.000	0.000	0.280	0.060
GA-DC	0.740	0.480	0.180	0.000	0.580	0.580	0.010	0.000	0.420	0.000
GA-RTS	1.000	1.000	0.288	0.000	0.240	0.240	0.000	0.000	0.370	0.060

function	F6(2D)		F7(2D)		F6(3D)		F7(3D)		F8(2D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	0.000	0.000	0.321	0.000	0.000	0.000	0.214	0.000	0.238	0.020
L2GA-L2PS	0.000	0.000	0.107	0.000	0.000	0.000	0.016	0.000	0.062	0.000
L2GA	0.000	0.000	0.028	0.000	0.000	0.000	0.005	0.000	0.053	0.000
L2PSO	0.383	0.000	0.315	0.000	0.196	0.000	0.131	0.000	1.000	1.000
GA-C	0.000	0.000	0.382	0.000	0.000	0.000	0.603	0.000	0.243	0.020
GA-DC	0.000	0.000	0.028	0.000	0.000	0.000	0.005	0.000	0.068	0.000
GA-RTS	0.000	0.000	0.105	0.000	0.000	0.000	0.022	0.000	0.123	0.000

function	F9(2D)		F10(2D)		F11(2D)		F11(3D)		F12(3D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2GA-L2PS	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2GA	0.007	0.000	0.005	0.000	0.003	0.000	0.007	0.000	0.028	0.000
L2PSO	0.513	0.000	0.455	0.000	0.503	0.000	0.670	0.000	0.670	0.000
GA-C	0.007	0.000	0.000	0.000	0.007	0.000	0.007	0.000	0.020	0.000
GA-DC	0.007	0.000	0.003	0.000	0.007	0.000	0.017	0.000	0.015	0.000
GA-RTS	0.003	0.000	0.000	0.000	0.007	0.000	0.003	0.000	0.005	0.000

function	F11(5D)		F12(5D)		F11(10D)		F12(10D)		F12(20D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2GA-L2PS	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2GA	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2PSO	0.667	0.000	0.487	0.000	0.027	0.000	0.000	0.000	0.000	0.000
GA-C	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
GA-DC	0.007	0.000	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000
GA-RTS	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Results per accuracy = 1E-05

function	F1(1D)		F2(1D)		F3(1D)		F4(2D)		F5(2D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	1.000	1.000	0.132	0.000	0.020	0.020	0.000	0.000	0.020	0.000
L2GA-L2PS	1.000	1.000	0.136	0.000	0.080	0.080	0.000	0.000	0.000	0.000
L2GA	0.620	0.240	0.104	0.000	0.180	0.180	0.000	0.000	0.210	0.000
L2PSO	0.910	0.840	0.656	0.340	1.000	1.000	1.000	1.000	1.000	1.000
GA-C	1.000	1.000	0.148	0.000	0.100	0.100	0.000	0.000	0.030	0.000
GA-DC	0.720	0.440	0.096	0.000	0.140	0.140	0.000	0.000	0.200	0.000
GA-RTS	1.000	1.000	0.100	0.000	0.080	0.080	0.000	0.000	0.070	0.000

function	F6(2D)		F7(2D)		F6(3D)		F7(3D)		F8(2D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	0.000	0.000	0.092	0.000	0.000	0.000	0.084	0.000	0.045	0.000
L2GA-L2PS	0.000	0.000	0.018	0.000	0.000	0.000	0.001	0.000	0.008	0.000
L2GA	0.000	0.000	0.027	0.000	0.000	0.000	0.005	0.000	0.010	0.000
L2PSO	0.000	0.000	0.271	0.000	0.140	0.000	0.117	0.000	1.000	1.000
GA-C	0.000	0.000	0.128	0.000	0.000	0.000	0.170	0.000	0.025	0.000
GA-DC	0.000	0.000	0.027	0.000	0.000	0.000	0.005	0.000	0.022	0.000
GA-RTS	0.000	0.000	0.059	0.000	0.000	0.000	0.019	0.000	0.023	0.000

function	F9(2D)		F10(2D)		F11(2D)		F11(3D)		F12(3D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2GA-L2PS	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2GA	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2PSO	0.500	0.000	0.417	0.000	0.500	0.000	0.670	0.000	0.665	0.000
GA-C	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
GA-DC	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.000	0.003	0.000
GA-RTS	0.003	0.000	0.000	0.000	0.000	0.000	0.003	0.000	0.000	0.000

function	F11(5D)		F12(5D)		F11(10D)		F12(10D)		F12(20D)	
algorithm	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
GA-L2PS	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2GA-L2PS	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2GA	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
L2PSO	0.667	0.000	0.487	0.000	0.000	0.000	0.000	0.000	0.000	0.000
GA-C	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
GA-DC	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
GA-RTS	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000