Applied Soft Computing 28 (2015) 379–393

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Solving molecular flexible docking problems with metaheuristics: A comparative study

Esteban López-Camacho, María Jesús García Godoy, José García-Nieto, Antonio J. Nebro*, José F. Aldana-Montes*

Lenguajes y Ciencias de la Computación, Universidad de Málaga, Bulevar Louis Pasteur 35, Málaga, Spain

ARTICLE INFO

Article history:
Received 20 June 2013
Received in revised form
11 September 2014
Accepted 10 October 2014
Available online 19 December 2014

Keywords:
Molecular docking
Optimization
Metaheuristics
Experimental comparison

ABSTRACT

The main objective of the molecular docking problem is to find a conformation between a small molecule (ligand) and a receptor molecule with minimum binding energy. The quality of the docking score depends on two factors: the scoring function and the search method being used to find the lowest binding energy solution. In this context, AutoDock 4.2 is a popular C++ software package in the bioinformatics community providing both elements, including two genetic algorithms, one of them endowed with a local search strategy. This paper principally focuses on the search techniques for solving the docking problem. In using the AutoDock 4.2 scoring function, the approach in this study is twofold. On the one hand, a number of four metaheuristic techniques are analyzed within an extensive set of docking problems, looking for the best technique according to the quality of the binding energy solutions. These techniques are thoroughly evaluated and also compared with popular well-known docking algorithms in AutoDock 4.2. The metaheuristics selected are: generational and a steady-state Genetic Algorithm, Differential Evolution, and Particle Swarm Optimization. On the other hand, a C++ version of the jMetal optimization framework has been integrated inside AutoDock 4.2, so that all the algorithms included in jMetal are readily available to solve docking problems. The experiments reveal that Differential Evolution obtains the best overall results, even outperforming other existing algorithms specifically designed for molecular docking.

1. Introductionintroduction

Molecular docking tools are used to predict the conformation and relative orientation of two or more molecular constituents. Pioneered in the early 1980s, these computational approaches have made an important contribution to the location and characterization of binding sites in macromolecules, as well as in the discovery, design and comparison of new drugs to find new compounds with high affinity to the targeted proteins [1].

The objective of docking is to obtain a model of interaction between a ligand and a macromolecule, characterized by a minimum binding energy. This model is based on known ligand and macromolecule 3D structures. In the past, this requirement was a problem due to the lack of 3D crystal structures. However, with the development of techniques such as high-throughput protein purification, the X-ray crystallography and nuclear magnetic resonance (NMR) spectroscopy, the number of structures stored in the Protein Data Bank database (PDB) [2] has increased. This database currently contains 100,547 entries for 3D structures obtained by X-ray crystallography (over 80%), NMR (about 16 %) and theoretical modeling (2%). Most of these PDB structures have an important function in the metabolic and biosignaling pathways and can be considered as drug targets in docking analysis. In addition, the advances in human genome sequencing have provided a basis for the detection of 3D protein structures related to the regulation and the expression of genes. In this context, these 3D structures offer a high diversity of biologically active macromolecules which are used to generate drug interaction models for the discovery of new targets.

In the computational development of docking software, researchers in this field have traditionally focused on two of the components which determine the quality of the docking software: the scoring function and the optimization algorithm. The scoring function enables the evaluation of conformations between ligand and protein, to detect those conformations with the minimum binding energy. Moreover, the conformational model used in

* Corresponding authors. Tel.: +34 952133310.
E-mail addresses: esteban@lcc.uma.es (E. López-Camacho), mjgarciag@lcc.uma.es (M.J. García Godoy), jnieto@lcc.uma.es (J. García-Nieto), antonio@lcc.uma.es (A.J. Nebro), jfam@lcc.uma.es (J.F. Aldana-Montes).

http://dx.doi.org/10.1016/j.asoc.2014.10.049
1568-4946/

docking software tools allows the rotation around torsional degrees of freedom of the flexible ligand and side chains of receptor in order to create more realistic docking simulations [3]. So, given a scoring function, the goal is to use an optimization algorithm to find the best docking solutions which are those complexes with the minimum binding energy. In this paper, we focus on metaheuristics [4], a broad family of non-exact optimization techniques, including Evolutionary Algorithms (EAs), Particle Swarm Optimization (PSO) [5], Ant Colony Optimization (ACO) [6], Simulated Annealing (SA), Tabu Search (TS), and many others. Genetic Algorithms (GAs), a subfamily of EAs, are the most well-known and used metaheuristics.

A popular tool for molecular docking is AutoDock [7], a C++ software package which in its 4.2 release provides a SA and two GAs algorithms, one of which, referred to as the Lamarckian GA, incorporates a local search [8]. These techniques involve specific operators using problem information, which leads them to outstanding results for rigid and flexible molecular confirmations. The main motivation in this study is to analyze the performance of a set of general-purpose metaheuristics (in their canonical design) to determine whether any of them can lead to better scoring values compared to the techniques already provided by AutoDock. In particular, we focus on two variants of standard GA (generational and steady-state), a PSO algorithm, and a Differential Evolution (DE) solver [9]. To achieve this goal, instead of trying to incorporate the new algorithms into the source code of AutoDock, the approach has been, to use a software framework oriented to solving optimization problems with metaheuristics. Specifically, the framework used is jMetal [10], a Java-based object-oriented software library aimed at multi-objective optimization but it also incorporates a number of single-objective algorithms. As AutoDock and jMetal are written in different programming languages, to obtain an efficient implementation we have opted to develop a C++ version of jMetal. The result is a software package that integrates both tools [11], in such a way that the algorithms available in jMetal can be used to optimize the binding energy function of AutoDock 4.2. Consequently, AutoDock users also benefit because they have the choice of using different optimization techniques beyond the ones already incorporated inside the software, and researchers in metaheuristics can use a real world problem, like molecular docking in their work.

For the algorithmic evaluation and comparison, a thorough experimentation is performed in this study, where algorithms are subjected to a benchmark of molecules with different properties of flexibility, size and resolution. As mentioned, the flexibility in macromolecule and ligands is considered a more realistic docking problem and also a challenge given the more complexity of the problem, therefore, we have chosen to focus here on solving this problem. The ligands are of different sizes from small to large including cyclic urea ligands.

The main contributions of this work can be summarized as follows:

- Four metaheuristics (two GAs, PSO, and DE) are used to solve molecular docking problems and their behavior analyzed in terms of performance. In addition, these algorithms are also compared with two well-known metaheuristics included in AutoDock 4.2 (GA and LGA).
- A set of 83 docking instances is generated based on real existing molecules, then constituting an extensive benchmark for assessing algorithms. From this set, 75 PDB structures (instances) are used for performance assessment involving flexibility in HIV-protease macromolecules and ligands, whereas a subset of 11 molecular instances are used for the parameter setting (three instances are used in both subsets), revealing new configurations for fine-tuning algorithms in the context of flexible molecular docking. The benchmark is online available and adapted to be reproduced using compatible software.[1] Along this paper, all necessary information is provided to make this study replicable according to standard guidelines [12].
- A thorough experimentation with statistical assessment of results is performed to compare algorithms. An additional analysis in terms of convergence behavior is also provided.
- A series of relevant solutions are analyzed from the point of view of their biological meaning, so as to offer insightful results for experts in the domain of application of molecular modeling and design.

The remainder of this paper is organized as follows. Section 2 includes a review of the state of the art concerning other approaches to the problem being tackled here. Section 3 presents the molecular docking problem, with solution encoding and fitness energy function formulation. In Section 4, the metaheuristics evaluated in this study are described. Section 5 includes the description of the strategy followed, based on the jMetal + AutoDock integration. Section 6 reports the experimental setup, with parameter settings and problem instances. Results, comparisons, and analysis are presented in Section 7. Section 8 concludes the paper and presents an overview of future work.

2. Related work

Over the last two decades, different metaheuristics have been applied as search methods to solve the docking problem [13]. One example is the docking software AutoDock, which incorporates three metaheuristic techniques. AutoDock is considered to be the most cited and one of the most used software packages in molecular modeling studies [7], because it is an efficient tool for virtual drug screening [14].

AutoDock was released in 1990, and it included a rapid search method using Monte Carlo simulated annealing [15]. However, this method proved to be inadequate for ligands with more than eight rotatable bonds [8]. Eight years later, in an attempt to improve the software, AutoDock 3.0 was released, adding the Genetic Algorithm (GA) and the Lamarckian Genetic Algorithm (LGA), which incorporates a local search, and an empirical binding free energy force that enables the prediction of binding of free energies. Docking analyses have demonstrated that the LGA is the most efficient search method of the three AutoDock algorithms in terms of the lowest energy found in a number of energy function evaluations [8]. AutoDock 4 was presented in 2009 [3]. It allows conformational models of side chains of proteins, provides torsional degrees of freedom and tries to solve the problem of flexibility in the receptor, a challenge in docking approaches. More recently, a new release has appeared, AutoDock 4.2, which incorporates several enhancements over AutoDock 4. The latest version includes a default unbonded state, different to the extended unbonded state of AutoDock 4, an improvement over the time required to run a high-quality docking with flexible and rigid components, involving an attempt to ensure compatibility between the different releases of AutoDock software.

In 2010, as an improvement on the previous releases, the AutoDock authors implemented a new program for molecular docking called AutoDock Vina [16], containing differences with respect to the previous versions, such as the energy function, multithreading to speed up the execution in multicore processors, and the use of an iterated local search algorithm (ILS) as the search engine. The stopping condition of the ILS is adaptively determined, thus making it difficult to compare it fairly with other techniques that use a fixed number of function evaluations.

[1] http://khaos.uma.es/AutodockjMetal/instances.jsp.

A number of approaches can be found in the current literature that have proposed metaheuristic techniques designed around AutoDock versions. Atilgan et al. [17] developed a new program named AutoDockX which incorporates a sustainable GA, namely Age-Layered Population Structure (ALPS), including the age attribute for individuals. Chen et al. [18] presented an algorithm called SODOCK, which is an adaptation of PSO including Solis and Wets local search and uses the energy function of AutoDock 3.05. Two other PSO related proposals are the varCPSO-ls algorithm, an extension of the CPSO algorithm with a local optimizer which is embedded inside the AutoDock 3 source code and uses its energy function [19], and the FIPSDock algorithm, which adopts the AutoDock 4.2 energy function [20]. DE has also been applied in this context. A first attempt is DockDE [21], a variant of DE which uses a older version of the AutoDock energy function. ODE is also an extension of DE enhanced by a local search algorithm and a pseudo-elitism operator, and using the AutoDock scoring function [22]. A more recent version is SADock [23], that incorporates a Hooke Jeeves local search.

Among studies on molecular docking with metaheuristics not based on AutoDock, there are several approaches that are also worth mentioning. PARADockS is a framework implemented to predict the ligand–protein interaction adapting PSO to several objective functions [24]. An Ant Colony Optimization approach is also presented in [25] using a systematic molecular simulator. A variant of DE called MolDock was parallelized on both GPU and CPU using a fitness function designed by the authors [26]. However, although the technique is adapted to a flexible docking receptor, it has not been evaluated using flexible targets. Other optimization methods such as multi-scale optimization models and information entropy-based searching techniques with narrowing space were applied in a new docking algorithm [27]. Herberlé et al. [28] review EAs applied to *Mycobacterium tuberculosis* docking targets.

In terms of analysing the influence of algorithm operators and parameters, a study developed by Thomsen [21] compared the performance of the LGA and DockEA algorithms by selecting different EA operators, populations and usage of local search. However, the parameter setting study proposed, included very few docking problems and was only applied to the DockEA algorithm. Another interesting study performed by Tavares et al. [29] investigated the effects of Gaussian and Cauchy mutation operators through a locality analysis (small genotype variations imply small variations in phenotype); the results showed that Gaussian-based operators had a stronger locality than Cauchy-based operators. They also demonstrated that the results of runs using the Gaussian-based operator were better than those returned by the Cauchy-based operator.

Some studies have been developed over the last five years based on multi-objective optimization, such as the reviews on the multi-objective application in different fields proposed by Nicolaou et al. [30] and Nicolotti et al. [31]; a new hybrid algorithm proposed by Grosdidier et al. [32] which uses two fitness functions where two variables are optimized; Sandoval-Perez et al. [33] used the NSGA-II algorithm implemented in the jMetal framework in order to optimize two variables (bonded and non-bonded energy terms); Oduguwa et al. [34] applied the NSGA-II, PAES, SPEA evolutionary multi-objective techniques optimizing up to three objectives of the energy function; and finally, Janson et al. [35] applied a multi-objective optimization approach using the PSO algorithm.

All these approaches have addressed different aspects of the molecular docking optimization. However, three common features can be found in most of them which clearly differentiates them from the main focus in this paper:

- Our study focuses on carrying out a comparative analysis with canonical metaheuristics in the solution of molecular docking problem. To the best of our knowledge, there have been no other studies performed which carry out this comparative analysis.
- Most of the work presented in this section uses a set with a low number of structures to be evaluated and concentrates on rigid macromolecules to be docked. The current study includes an extensive set of 83 molecular structures. In this set, receptors are HIV-proteases and ligands that have different sizes and flexibility applied to macromolecule and ligand. These problems of macromolecule flexibility are not considered in the current reviewed literature, therefore we are providing new fresh results for the scientific community.
- A systematic parameter setting with multiple combinations of parameter values statistically validated is performed here. In addition to all these contributions, a software framework has been developed,[2] which integrates AutoDock 4.2 and jMetal, enabling the community to use those algorithms that are not available in AutoDock.

3. The problem: molecular docking

The main objective in a molecular docking problem is to find an optimized conformation between the ligand (L), a rigid/flexible small molecule, and the receptor (R), a rigid/flexible macromolecule that results in a minimum binding energy. The interaction between ligand and receptor can be described by an objective function calculated according to three components representing degrees of freedom: (1) the translation of the ligand molecule, involving the three axis values (x, y, z) in Cartesian coordinate space; (2) the ligand orientation, modeled as a four variables quaternion including the angle slope (w); and (3) the flexibilities, represented by the free rotation of torsion (dihedral angles) of the ligand and sidechains of the receptor.

Solution encoding. As illustrated in Fig. 1, each problem solution for AutoDock and jMetal is encoded by a real-value vector of $n+7$ variables, in which the first three values correspond to the ligand translation, the next four values correspond to the ligand and/or macromolecule orientation, and the remaining n values are the ligand torsion dihedral angles. Nevertheless, with the aim of reducing the computational cost, a grid-based methodology has been implemented where the protein active site is embedded in a 3D rectangular grid, and on each point of the grid, the electrostatic interaction energy and the van der Waals terms for each ligand atom type are pre-computed and stored, taking into account all the protein atoms. In this way, the protein contribution at any given point is obtained by tri-linear interpolation in each grid cell. This interpolation leads to a range of translation variables (x, y, z) of 120 grid spacing points dimension (see [36]). The values of these variables are delimited between the range of the coordinates of the grid space that has been chosen for each problem. All ranges are selected randomly, so if the center of the grid is for example the (10, 10, 10) point, a solution with values of ten in its x, y and z will be in such a position. In the case of orientation (quaternion) and torsion variables, they are measured in radians and encoded in the range of $[-\pi, \pi]$.

Fitness function. In this approach, the energy scoring function (or fitness function) recommended in AutoDock 4.2 [36] is used to measure the quality of binding in solutions, as it was empirically determined in the scope of this tool. This scoring function, also called semi-empirical energy force field, computes the total free binding energy of the ligand-macromolecule complex according to the following equations:

[2] Available at URL: http://khaos.uma.es/AutodockjMetal/.

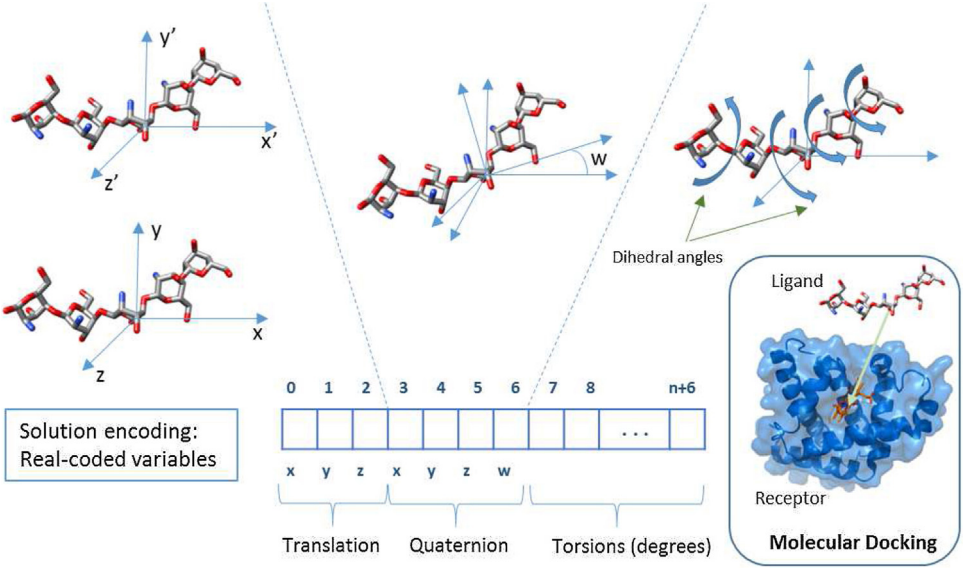

Fig. 1. Solution encoding in AutoDock 4.2 and jMetal. The first three values (translation) are the coordinates of the center of rotation of the ligand. The next four values (quaternion) are the unit vector describing the direction of rigid body rotation (x, y and z) and the rotation of the angle degrees (w) that are applied. The rest of the values hold the torsion angles in degrees, being n the number of torsions of the ligand.

$$\Delta G = (V^{L-L}_{\text{bonded}} - V^{L-L}_{\text{unbonded}}) + (V^{R-R}_{\text{bonded}} - V^{R-R}_{\text{unbonded}}) + (V^{R-L}_{\text{bonded}} - V^{R-L}_{\text{unbonded}} + \Delta G_{\text{conf}}) \quad (1)$$

$$V = W_{vdw}\sum_{i,j}\left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}}\right) + W_{hbond}\sum_{i,j}E(t)\left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}}\right) + + W_{elec}\sum_{i,j}\frac{q_i q_j}{\varepsilon(r_{ij})r_{ij}} + W_{sol}\sum_{i,j}(S_i V_j + S_j V_i)e^{(-r_{ij}^2/2\sigma^2)} \quad (2)$$

$$\Delta G_{conf} = W_{conf} N_{tors} \quad (3)$$

As formulated in Eq. (1), the free binding energy function is calculated from the differences between ligand/s (L) and receptor/s (R) in bonded and unbonded states. That is, the free energy of binding is based on the evaluation of the transition of the ligand and protein intramolecular energetics from an unbonded to a bonded state, and the intermolecular energetics of ligand–protein complex. Therefore, the force field involves six pair-wise evaluations (V) plus a term of conformational entropy (N_{tors}), which is directly proportional to the number of rotatable bonds of the ligand molecule (Eq. (3)). Each pair of energetic evaluation terms includes evaluations of dispersion/repulsion (*vdw*), hydrogen bonds (*hbond*), electrostatics (*elec*), and desolvation (*sol*). Weights W_{vdw}, W_{hbond}, W_{conf}, W_{elec}, and W_{sol} of Eqs. (2) and (3), are constants for van der Waals, hydrogen bonds, torsional forces, electrostatic interactions, and desolvation, respectively. In Eq. (2), r_{ij} represents the interatomic distance, A_{ij} and B_{ij} in the first term are Lennard–Jones parameters taken from Amber force field [37]. Similarly, C_{ij} and D_{ij} in the second term are Lennard-Jones parameters for maximum well depth of potential energies between two atoms, and $E(t)$ represents the angle-dependent directionality. The third term in Eq. (2) uses a Coulomb approach for electrostatics. Finally, the fourth term is calculated from the volume (V) of the atoms that are surrounding a given atom weighted by S, and an exponential term which involves atom distances.

Ligand–protein docking is a highly complex optimization problem, with unknown optimum and usually characterized by multimodal landscape energy functions [38]. In addition, the computational cost of each energy evaluation increases with the number of atoms in complex ligand-protein (with thousands of them), hence involving millions of energy evaluations, since a minimum quality of molecular binding is mandatory in chemistry research. Therefore, the use of metaheuristic approaches is highly recommendable for molecular docking, since they are able to explore a great number of combinations with a fast convergence to successful solutions [4].

4. Metaheuristics

This section describes the four metaheuristic algorithms evaluated in this study. Specifically, they are two evolutionary algorithms, steady state Genetic Algorithm (ssGA) and generational Genetic Algorithm (gGA); and two differential vector techniques, Particle Swarm Optimization (PSO), and Differential Evolution (DE). These techniques have been selected as they are widely used in the literature to solve real-coded optimization problems [39]. In addition, they lead to experiments with different population structures and reproduction mechanisms.

4.1. Genetic Algorithms

Genetic Algorithms [40] are the most popular metaheuristic algorithms belonging to the subfamily of Evolutionary Algorithms (EAs). A GA iterates a process in which two solutions (parents) are selected from the whole population with a given selection criterion. These parents are then recombined to obtain offspring solutions. The offsprings are mutated and finally they are evaluated and inserted back into the population following a given replacement criterion. As the recombination operator, a simulated binary crossover (SBX) is used for continuous variables [39]. The mutation process is carried out by selecting one of the elements in the solution, and assigning a new value in the range of problem variables. Algorithm 1 summarizes the operations of a canonical GA.

Algorithm 1. Pseudocode of GA

```
1: P^0 ← initializePopulation()
2: while g < MAXIMUM_g do
3:   R^g ← recombine(P^g)
4:   M^g ← mutate(R^g)
5:   evaluate(M^g)
6:   P^{g+1} ← select(M^g ∪ R^g)
7: end while
```

There are two main versions of GA: *steady state* GA (ssGA) and *generational* GA (gGA). The difference between the ssGA and the gGA is the way in which the population is being updated with the new individuals generated during the evolution. In the case of ssGA, the new individuals are directly inserted into the current population, whereas in the case of the gGA, a new auxiliary population is built with the offsprings obtained and then, once this auxiliary population is full, it replaces the current population. Thus, in ssGAs the population is being asynchronously updated with the newly generated individuals, while in the case of genGAs all the new individuals are updated at the same time, in a synchronous way.

It is worth mentioning that jMetal [41] incorporates a number of GAs, including steady-state (ssGA), generational (gGA), as well as several cellular GA variants. In this approach, we have selected the ssGA and the gGA, which have been configured with common settings used in many approaches [39,42]: binary tournament as the selection operator, and polynomial mutation and simulated binary crossover (SBX). All these operators work without any specific knowledge of the problem.

In addition, AutoDock 4.2 also incorporates a GA following a steady-state selection scheme, where the degree of elitism (i.e., the number of best solutions that survive into the next generation) is user-defined. The evolutionary operators are proportional selection, 2-point crossover, and Cauchy mutation. The crossover operators are adapted to the problem solution structure, as the solution is never split using a quaternion value position. Furthermore, a hybrid version of GA with local search, the aforementioned Lamarckian GA (LGA), is also provided in AutoDock 4.2. The local search method follows a pseudo-Solis–Wets scheme [43], and consists of a number of iterations during which rotational, translational, and torsional degrees of freedom of the ligand are randomly sampled at a given step. These last two GAs are also compared with the four selected metaheuristics in the experimental sections of this study.

4.2. Particle Swarm Optimization

Particle Swarm Optimization [5] is a population based metaheuristic inspired by the social behavior of birds within a flock, and was initially designed for continuous optimization problems. In this algorithm, each particle position $\mathbf{x}_i$ (solution) is updated each generation t by means of Eq. (4).

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{4}$$

where $\mathbf{v}_i^{t+1}$ is the velocity vector of the particle given by

$$\mathbf{v}_i^{t+1} = \chi(\mathbf{v}_i^t + U^t[0, \varphi_1] \cdot (\mathbf{p}_i^t - \mathbf{x}_i^t) + U^t[0, \varphi_2] \cdot (\mathbf{b}_i^t - \mathbf{x}_i^t)) \tag{5}$$

In this formula, $\mathbf{p}_i^t$ is the personal best position the particle i has ever stored, $\mathbf{b}_i^t$ is the position found by the member of its neighborhood that has had the best performance so far. Acceleration coefficients φ_1 and φ_2 control the relative effect of the personal and social best particles, and U^t is a diagonal matrix with elements distributed in the interval $[0, \varphi_i]$, uniformly at random. Finally, χ is the Clerc's constriction coefficient [44]. This coefficient is calculated by means of Eq. (6), from the two acceleration coefficients φ_1 and φ_2, being the sum of these two coefficients that determines the χ to use. Usually, $\varphi_1 = \varphi_2 = 2.05$, giving as results $\varphi = 4.1$, and $\chi = 0.7298$ [44]

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \quad with\ \varphi = \sum_i \varphi_i,\ and\ \varphi > 4 \tag{6}$$

Algorithm 2. Pseudocode of PSO

```
1: S ← initializeSwarm()
2: while t < MAXIMUM_t do
3:   for each particle x_i^t do
4:     updateVelocity(v_i^t) //Eq. (5)
5:     updatePosition(x_i^t) // Eq. (4)
6:     evaluate(x_i^t)
7:     update(p_i^t)
8:   end for
9:   updateLeader(b_i^t)
10: end while
```

Algorithm 2 describes the pseudo-code of PSO. In concrete, this PSO specification is based on the canonical version proposed by Cler's [44], for which only one specific parameter is required to be tuned (φ). The algorithm starts by initializing the Swarm (Population) which includes both the positions and velocities of the particles. The corresponding $\mathbf{p}_i$ of each particle is randomly (uniformly) initialized, as well as the leader $\mathbf{b}$. Then, during a maximum number of iterations, each particle *flies* through the search space updating its velocity and position (Lines 4 and 5), it is then evaluated (Line 6), and its $\mathbf{p}_i$ is also calculated (Line 7). At the end of each iteration, the leader $\mathbf{b}$ is updated.

4.3. Differential Evolution

Differential Evolution [9] is a stochastic population based algorithm designed to solve optimization problems in continuous domains. The task of generating new individuals is performed by differential operators such as the differential mutation and crossover. A *mutant individual* $\mathbf{w}_i^{t+1}$ is generated by the following Eq. (7):

$$\mathbf{w}_i^{t+1} \leftarrow \mathbf{x}_{r1}^t + \mu \cdot (\mathbf{x}_{r2}^t - \mathbf{x}_{r3}^t) \tag{7}$$

where $r1, r2, r3, r4 \in \{1, 2, \ldots, i-1, i+1, \ldots, N\}$ are random integers mutually different, and also different from the index i. The mutation constant $\mu > 0$ stands for the amplification of the difference between the individuals $x_{r's}$ and it avoids the stagnation of the search process. Each mutated individual undergoes a crossover operation with the *target individual* x_i^t, by means of which a *trial individual* u_i^{t+1} is generated. A randomly chosen position is taken from the mutant individual to prevent that the trial individual replicating the target individual.

$$u_i^{t+1}(j) \leftarrow \begin{cases} w_i^{t+1}(j) & \text{if } r(j) \le Cr \text{ or } j = j_r, \\ x_i^t(j) & \text{otherwise.} \end{cases} \tag{8}$$

As shown in Eq. (8), the crossover operator randomly chooses a uniformly distributed integer value j_r and a random real number $r \in [0, 1]$, also uniformly distributed for each component j of the trial individual u_i^{t+1}. Then, the crossover probability Cr, and r are compared just like j and j_r. If r is lower or equal than Cr (or j is equal to j_r) then we select the jth element of the mutant individual to be allocated in the jth element of the trial individual u_i^{t+1}. Otherwise, the jth element of the target individual x_i^t becomes the jth element of the trial individual.

Finally, a selection operator decides on the acceptance of the trial individual for the next generation if and only if it yields a

reduction (assuming minimization) in the value of the fitness function $f()$, as shown by the following Eq. (9):

$$x_i^{t+1} \leftarrow \begin{cases} u_i^{t+1} & \text{if } f(u^{t+1}i) \leq f(x_i^t), \\ x_i^t & \text{otherwise.} \end{cases} \quad (9)$$

Algorithm 3 shows the pseudocode of DE. After initializing the population, the individuals evolve during a number of iterations ($MAXIMUM_t$). Each individual is then mutated (Line 5) and recombined (Line 6). The new individual is selected (or not) following the operation of Eq. (9) (Lines 7 and 8).

Algorithm 3. Pseudocode of DE

```
1:  P←initializePopulation(Ps)
2:  while t < MAXIMUM_t do
3:    for each individual i of P do
4:      choose mutually different r_s values
5:      w_i^{t+1} ← mutation(x_{r_s}^t, μ) //Eq. (7)
6:      u_i^{t+1} ← crossover(x_i^t, w_i^{t+1}, Cr) //Eq. (8)
7:      evaluate(u_i^{t+1})
8:      x_i^{t+1} ← selection(x_i^t, u_i^{t+1}) //Equation 9
9:    end for
10: end while
```

5. Optimization strategy: jMetal + AutoDock

The proposed optimization strategy is composed of two main processes acting in parallel: an optimization algorithm and a molecular docking procedure. The optimization part is carried out through a metaheuristic algorithm (ssGA, gGA, PSO, or DE) provided by jMetal [41]. jMetal is an open-source framework intended for single/multi-objective optimization metaheuristics. The molecular binding procedure is performed by AutoDock 4.2 [36], a C++ tool for virtual drug discovery widely used for real cases which involves rigid, as well as flexible docking.

The proposed strategy implements an integration between jMetal and AutoDock in such a way that the algorithms available in jMetal can communicate with AutoDock to cooperate on the molecular docking optimization. To carry out this integration, jMetal framework was translated to C++ from scratch. This new version in C++ was called jMetalCpp. Such a solution consists of running AutoDock and jMetalCpp in two different threads inside the same process, memory is therefore shared so they can communicate with each other and they synchronize using mutexes. This approach is efficient and flexible, allowing any of the algorithms included in jMetalCpp to be easily used for solving docking problems.

In this way, as illustrated in Fig. 2, when a given metaheuristic algorithm in jMetal is executed, it performs the optimization procedure and generates new solutions. Whenever the binding quality of a new solution has to be numerically quantified, it is sent to AutoDock to be evaluated with the binding energy function (Eq. (1)). After this evaluation, AutoDock returns the corresponding binding energy to the algorithm so it can assign this fitness value to the evaluated solution. This process is repeated until reaching a preestablished stop condition. As a result, the best solution found so far by the metaheuristic is returned to AutoDock to generate the output data concerning the optimized docking. Thus, the final results follow the standard format tailored to AutoDock users.

6. Experimental setup

This section describes the sets of molecular (ligand–receptor) problem instances used for optimal docking in this study. Then, the methodology for preparing docking instances and software packages is also explained. Finally, the parameter settings of the evaluated algorithms are described.

6.1. Molecular docking problem instances

The experimental procedure followed in this study comprises two different sets of protein–ligand complexes. Both sets of molecules were taken from the PDB database [45], and they are available online for experimental reproduction. The first set consists of 11 molecules with flexible ligands and receptors. This set has been used to carry out the parameter setting of algorithms (DE, PSO, ssGA, and gGA) evaluated in multiple systematic docking simulations. The second set comprises 75 molecules, also with flexible ligands and macromolecules, and is used for experimental comparisons. Specifically, the first set of molecules is used to train the learning procedure induced by tuning parameters in algorithms, whereas the second set is used to test and validate the actual performance of selected metaheuristics on a great number of docking instances.

Table 1 summarizes the first set of selected problems showing the X-ray crystal structures, the PDB accession code, the structure resolution, and the reference where they were initially analyzed. In concrete, this first set of instances (1AJV, 1AJX, 1D4K, 1G2K, 1HIV, 1HPX, 1HTF, 1HTG, 1HVH, 1VB9, and 2UPJ) has been used for parameter settings, since they constitute a varied number of complexes with heterogeneous features and sizes. In addition, these instances have been previously used in [3] where the macromolecule flexibility is incorporated in the energy function of AutoDock 4.2.

According to the size and type of ligand, 1HPX, 1HIV, 1HTG and 1D4K complexes include large inhibitors (the largest being ligand 1HPX and the smallest, 1D4K); the ligand of 1HTF structure is categorized as a small ligand and 1AJX, 1AJV, 2UPJ, 1HVH and 1G2K complexes include cyclic urea inhibitors. In general, the ligands' torsional degrees of freedom are limited to 10, selecting those torsions that allow the fewest number of atoms to move around the ligand core. For the flexibility in the receptor, previous docking studies [3] with this set have demonstrated that residue ARG-8 showed the highest number of bad atom contacts with the ligand. Therefore, the flexibility docking was carried out adding three degrees of freedom in ARG-8, in both A and B receptor chains.

The second set consists of an extensive number of complexes (75) used by [3]. Similarly to the first set, the flexibility of ligand of this second set was limited to 10 torsional degrees of freedom, and the macromolecule flexibility is performed adding three degrees of freedom in ARG-8 residue in both side chains, A and B. This aminoacid was selected due to previous docking studies in which these complexes showed the highest number of bad atom contacts with the ligand with resolution (Å) ranges between 1.09 and 2.8 (more details in [3]). As shown in Table 2, this second group of complexes can be classified in terms of type of ligand, from small to large size inhibitors, and cyclic urea inhibitors. The set of structures classified according to the ligand size, the PDB code, and the range of crystalographic resolution in Å are also shown in Table 2.

In general, a total number of 83 structures have been used in experiments, since three instances in the set of 11 molecules are also used in the set of 75. Therefore, the parameter setting experiments steer the algorithms being compared to sources of knowledge, both dependent and independent. These sets of problems are complex and widely demanded by the docking user community, although it constitutes an extensive and varied enough set to test the compared techniques from an algorithmic point of view.

6.2. Carrying out the docking experiments

Solving the docking experiments with AutoDock and jMetal requires a number of steps that are enumerated here:

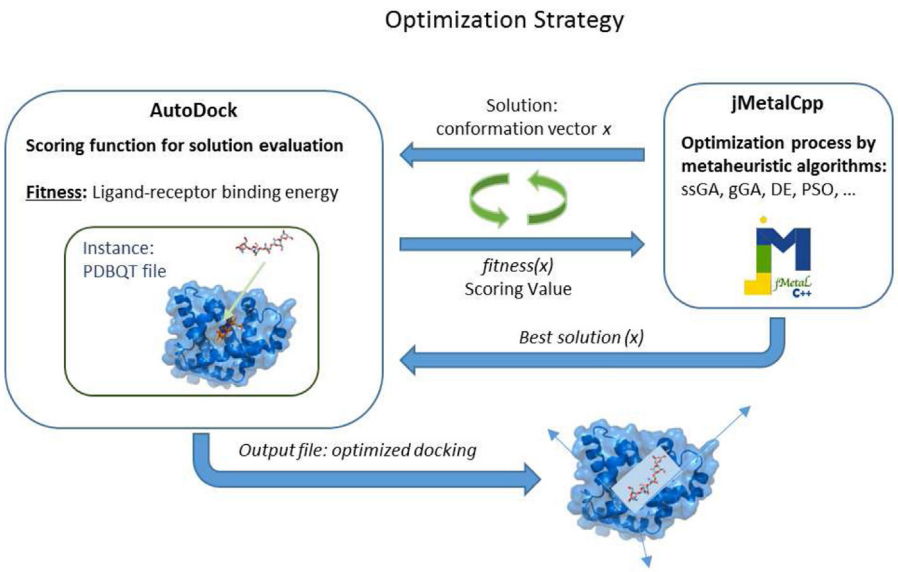

Fig. 2. This schema shows the integration between AutoDock and jMetalCpp. Solutions generated by jMetalCpp framework are evaluated by AutoDock and sent back to jMetalCpp. Once the stop condition has been met, the best solution is returned to the AutoDock code which in turn generates an output file with the results. These results can be visualized with any tool that is suitable for visualizing AutoDock format (DLG) like AutoDockTool (ADT).

Table 1
X-ray crystal structure coordinates taken from the PDB database and used for parameter settings, for flexible receptors. Their accession codes from the PDB database, resolution and references where they were initially analyzed are also presented.

Protein–ligand complexes	PDB	Resolution (Å)	References
HIV-1 protease/AHA006	1AJV	2	[46]
HIV-1 protease/AHA001	1AJX	2	[46]
HIV-1 protease/Macrocyclic peptidomimetic inhibitor 8	1D4K	1.85	[47]
HIV-1 protease/AHA047	1G2K	1.95	[48]
HIV-1 protease/U75875	1HIV	2	[49]
HIV-1 protease/KNI-272	1HPX	2	[50]
HIV-1 protease/GR126045	1HTF	2.20	[51]
HIV-1 protease/GR137615	1HTG	2	[51]
HIV-1 protease/Q8261	1HVH	1.80	[52]
HIV-1 protease/α-D-glucose	1VB9	2.20	[53]
HIV-1 protease/U100313	2UPJ	3	[54]

1. *Preparing the macromolecule and the ligand:* The X-ray crystallographic structures were downloaded from the PDB database [45]. Chimera UCSF software [55] was used to remove small molecules such as solvent molecules, non-interacting ions, waters, etc. from crystal structure. After this, the AutoDockTools (ADT) graphical interface suite [36] is used to prepare the macromolecule/ligand coordinate files. For the preparation of ligand PDBQT files, partial atomic charges and the AutoDock atom types (for each ligand atom) are computed and assigned. In the case of macromolecule PDBQT files, atom partial charges and hydrogens are added by using, respectively, Gasteiger and Babel methods [36]. The flexibility of the ligand is limited to 10 torsional degrees of freedom and six for the macromolecule. The rigid and flexible parts of HIV-proteases are specified and saved as corresponding rigid and flexible PDBQT files.
2. *Running Autogrid:* AutoDock simulations require a previous calculation of grid maps in order to reduce the acting area for ligand–receptor movements. This grid calculation means that optimization algorithms deal with a reduced search space, thus guiding the search to more promising areas. These maps are calculated through Autogrid [36], once the coordinates have been established (120Å x 120Å x 120Å and 0.375 Å of grid spacing) and centered on the co-crystallized ligand binding site. Once the Autogrid files have been obtained, a docking parameter file is configured to run AutoDock.

Table 2
X-ray crystal structure coordinates taken from the PDB database and used in docking experiments. They consist of 75 molecules with accession codes from the PDB database. The range of resolution (Å) of each subgroup is also shown in the last column.

Ligand type	PDB code	Resolution (Å)
Small size	1B6M, 1HTE, 1HPV, 1B6L, 1BDL, 1HEG, 1HIH, 1A9M, 1GNM, 3AID, 1HBV, 1GNN, 1HSG, 1AAQ, 1TCX, 1GNO, 1BDQ, 1BDR, 1SBG, 1KZK, 1Z1H, 1Z1R, 1Z1H	1.09–2.8
Medium size	5HVP, 1K6P, 1MUI, 1JLD, 1B6J, 1K6C, 1IZI, 4PHV, 1HEF, 1K6T, 1IZH, 1HXW, 1IIQ, 1HPS, 1B6P, 1B6K, 4HVP, 2BPX, 2BPV, 1D4L, 1K6V, 1MTR, 1D4K	1.75–2.8
Large size	1VIK, 1HIV, 1HVS, 8HVP, 1ODY, 1A94, 3TLH, 1HVI, 1HOS, 1VIJ, 9HVP, 7HVP, 1HVJ, 1HVK, 1HVL, 1HWR, 1HTG	1.8–2.8
Cyclic urea	7UPJ, 1QBT, 1BV7, 1QBR, 1HPO, 1MEU, 1G35, 1PRO, 1BWA, 1BWB, 1QBU, 1DMP, 1BV9, 1MES	1.8–2.5

3. *Running AutoDock + jMetal:* Using the configuration files created in the previous steps, AutoDock is executed for each docking instance (as explained in Section 5). Original metaheuristics provided by AutoDock (GA and LGA) are executed here by means of AutoDock 4.2. Evaluated algorithms included in the jMetal framework (gGA, ssGA, DE and PSO) are executed in the proposed optimization strategy jMetal + AutoDock. All algorithms used, including the four from jMetal and the two from AutoDock, have been executed in computers equipped with modern dual core processors, 2 GB RAM, and Windows 7 O.S. They operate under a Condor [56] middleware platform, managing a maximum number of 400 processors, that acts as a distributed task scheduler (each task dealing with one independent run). Finally, given that metaheuristics are stochastic techniques, 30 independent runs have been carried out per algorithm configuration and instance, in order to obtain statistical confidence in the context of the whole experimentation.

6.3. Parameter settings

In this study, parameters specific to each algorithm have been systematically tuned, whereas those similar ones involved in the experimental procedure have been commonly set, in order to establish comparisons as fairly as possible.

Concerning specific parameters, Table 3 shows the results obtained from executions with all parameter combinations considered for each algorithm, when solving the set of 11 molecular instances (Table 1) used for this purpose (training).[3] These results are then compared by means of a Friedman statistical test [57] the ranking values of which are shown in this table for each pair of parameters considered, and for each algorithm evaluated.

In this way, Genetic Algorithms in jMetal (gGA and ssGA) apply binary tournament selection, simulated binary crossover (SBX) with crossover probability $p_c = 0.9$, and polynomial mutation with mutation probability $p_m = 1/L$ (L is the number of problem variables). These two parameters (p_c and p_m) have been set with their standard values as proposed in the literature [39,42], leading to the systematic tuning for distribution indexes of crossover and mutation, η_c and η_m, respectively. Therefore, according to Table 3 (top left and right), distribution indexes resulted in values $\eta_c = 5$ and $\eta_m = 100$, in the case of gGA, and $\eta_c = 5$ and $\eta_m = 5$, in the case of ssGA. An interesting observation in this regard is that, low distribution indexes in SBX show the best performance for the two GAs in jMetal, therefore leading to an exploding behavior. In contrast, for gGA, the mutation distribution index is higher ($\eta_m = 100$), which induces a high diversity in the population, and then, an explorative search procedure.

In the case of DE (variant *rand/1/bin*), mutation constant μ and crossover probability Cr are tuned as shown in Table 3 (bottom left), with the result that the combination of $\mu = 0.5$ and $Cr = 0.9$ are the best performing parameters for this algorithm (Friedman's rank 2.59). Finally, for PSO, acceleration coefficients (φ_1 and φ_2) are tuned using all combinations of considered values (2.05, 2.10, 2.15, 2.20, and 2.25), performing a series of 30 independent runs for each one of the combinations. Therefore, as shown in Table 3 (bottom right), the best parameter combination found for PSO is $\varphi_1 = 2.05$ and $\varphi_2 = 2.05$, resulting in the lowest Friedman value (2.40) in this ranking. In addition, these results agree with the ones analytically set in the study of Clerc [44] concerning the PSO constriction factor.

Another interesting observation in Table 3 lies in the resulting p-values of statistical tests for tuned algorithm, which are lower than the confidence level ($\alpha = 0.05$) in almost all cases (excepting ssGA, although showing a low p-value = $2.01E-1$). Therefore, it is notable that selected combinations of values are diverse enough to cover a wide range of possibilities in the parameter setting, hence obtaining fine-tuned parameters.

Algorithms GA and LGA included in AutoDock 4.2 were set as done in the reference study in which they were proposed [8]. Both GAs apply 2-point crossover with a crossover probability $p_c = 0.8$ and Cauchy mutation with probability $p_m = 0.02$, $\alpha = 0.0$, and $\beta = 1.0$. The local search used in the LGA follows a pseudo Solis–Wets scheme with a maximum number of iterations of 300. Some parameters that are applied to both the algorithms in jMetal and AutoDock 4.2, such as the probabilities of crossover and mutation in the GAs, differ in their values (e.g., $p_c = 0.9$ in jMetal and $p_c = 0.8$ in AutoDock), in order to maintain the default values adopted in each framework instead of unifying them.

Table 4 summarizes the set of parameters selected to evaluate the algorithms. In this table, the population size is also included, which has been set to 150 individuals for all algorithms and instances. This population size is used as the default value in AutoDock [36], as in previous studies [58] where the same HIV-protease structures were tested. The number of energy function evaluations is 1,500,000 for all algorithms (including those from AutoDock4 docking simulations), then establishing a common stop condition for the experiments. In this regard, in the case of the instances of docking problems with flexible residues it is recommended that they be simulated using a large enough number of evaluations [3], so this value has been selected from preliminary trace observations (as shown in Fig. 4 and explained in next section), to assure convergence on a candidate solution, but avoiding taking an excessive time to do it.

7. Results

In this section, the performance behavior of the metaheuristics studied is analyzed, including numerical and statistical comparisons. In addition, a biological analysis of selected outstanding solutions is also provided for the experts in the problem domain.

7.1. Performance comparisons

Tables 5 and 6 show the median and best binding energies of distribution results, for the set (test) of 75 docking instances and for the six compared algorithms (gGA, ssGA, DE, and PSO from jMetal; GA and LGA from AutoDock), respectively. These binding scoring values are the energies in kcal/mol associated with the receptor–ligand complex, as previously specified (Section 6.1). Therefore, the lower the binding energy the better the result. Table 8 (in Section 7.2) contains the median RMSD (Root Mean Square Deviation) values associated with each energy binding value obtained. The RMSD is a measure of similarity between the experimental ligand position in the macromolecule and the computed position of the docking ligand. RMSD values are also shown in this study to allow further biological comparisons for the experts in the domain of molecular design.

In Table 5, the first observation that can be made is that DE (jMetal) shows the best median energy (marked with gray background) for the highest number of molecular instances. Specifically, DE obtained the best median conformation on 67 out of 75 receptor-ligand complexes, followed by the LGA (AutoDock) with 8 best medians (for molecules 1B6L, 1BDL, 1HEF, 1HIV, 1HPO, 1K6C, 1Z1H, and 1Z1R). In terms of the best binding energy, the DE algorithm also shows the highest number of best solutions (Table 6), achieving the lower binding energy in 37 out of 75 molecules, followed by the LGA with 35. In this regard, the remaining algorithms of jMetal showed the best binding energy in one molecule, obtaining the best solutions: gGA for 1BWB, ssGA for 1Z1H, and PSO for 7HVP.

In order to provide these results with statistical confidence (in this study $\alpha = 0.05$), a series of non-parametric statistical tests have been applied, as in several cases the distributions of results did not follow the conditions of normality and homoskedasticity [57]. Therefore, analyses and comparisons focus on the entire distribution of binding energies, although they pay particular attention to the Median values (Table 5), for the 75 tackled molecular instances. In particular, Friedman's ranking and Holm's post-hoc multicompare tests [57] have been applied in order to know which algorithms are statistically worse than the control one (the algorithm with the best ranking).

In this regard, as shown in Table 7, DE (jMetal) reaches the best ranking value (Friedman) with 1.10, followed by LGA, gGA, ssGA, GA, and PSO. Therefore, DE is

[3] In total, these tuning experiments consisted of: 4 (algorithms) × 25 (parameter combinations) × 30 (independent runs) × 11 (problem instances) = 33,000 tuning experiments.

Table 3
gGA, ssGA, DE and PSO parameter tuning: Friedman test of all parameter combinations, for each algorithm and for each selected molecular instance (confidence level $\alpha = 0.05$). Each test value in the tables has been calculated from the median distribution out of 30 independent runs.

gGA		η_m				
		5	20	60	**100**	160
	5	10.95	11.27	8.54	6.36	7.22
	20	12.54	12.56	14.45	13.09	10.72
η_c	60	15.22	13.36	16.09	13.22	15.09
	100	14.77	15.04	15.09	16.36	15.36
	160	14.04	12.86	12.54	16.13	14.95

According to χ^2 with 24 degrees of freedom: 37.73
With Friedman's p-value = 3.68E−2 (Rejects H_0)

DE		μ				
		0.1	0.3	**0.5**	0.7	0.9
	0.1	5.90	7.99	9.36	12.63	13.01
	0.3	4.54	8.27	13.27	15.90	19.31
Cr	0.5	12.09	5.54	12.72	19.00	22.99
	0.7	14.72	6.04	7.04	20.90	24.27
	0.9	19.09	7.27	2.59	16.63	23.81

According to χ^2 with 24 degrees of freedom: 204.76
With Friedman's p-value = 1.31E−10 (Rejects H_0)

ssGA		η_m				
		5	20	60	100	160
	5	7.99	8.36	12.86	13.68	13.50
	20	13.63	17.77	14.27	13.09	15.27
η_c	60	13.63	11.95	10.54	12.72	11.95
	100	10.95	10.86	14.63	16.13	14.36
	160	8.63	16.27	14.68	12.59	14.59

According to χ^2 with 24 degrees of freedom: 29.30
With Friedman's p-value = 2.01E−1 (Rejects H_0)

PSO		φ_2				
		2.05	2.10	2.15	2.20	2.25
	2.05	2.40	7.63	5.77	13.63	15.54
	2.10	7.09	7.63	12.45	14.81	14.27
φ_1	2.15	8.13	12.81	12.18	14.18	16.18
	2.20	17.36	12.72	16.72	18.72	16.36
	2.25	10.04	13.72	18.36	16.27	19.40

According to χ^2 with 24 degrees of freedom: 96.09
With Friedman's p-value = 2.23E−10 (Rejects H_0)

Table 4
Parameter settings of compared algorithms.

Framework	Algorithm	Parameter	Value
jMetal	gGA	Population size	150
		Selection	Binary tournament
		Crossover	SBX, $p_c = 0.9$, $\eta_c = 5$
		Mutation	Polynomial, $p_m = 1/L$, $\eta_m = 100$
	ssGA	Population size	150
		Selection	Binary tournament
		Crossover	SBX, $p_c = 0.9$, $\eta_c = 5$
		Mutation	Polynomial, $p_m = 1/L$, $\eta_m = 5$
	DE	Population size	150
		Differential crossover	$CR = 0.9$
		Scaling factor	$F = 0.5$
	PSO	Swarm size	150
		Acceleration coefficients	$C_1 = 2.05$, $C_2 = 2.05$
AutoDock 4.2	GA	Population size	150
		Crossover	2-point, $p_c = 0.8$
		Mutation	Cauchy, $p_m = 0.02$, $\alpha = 0.0$, $\beta = 1.0$
	LGA	Population size	150
		Crossover	2-point, $p_c = 0.8$
		Mutation	Cauchy, $p_m = 0.02$, $\alpha = 0.0$, $\beta = 1.0$
		LS scheme	Solis–Wets (300 iterations)

Table 5
Performance results of compared algorithms (gGA, ssGA, DE, and PSO from jMetal; GA and LGA from AutoDock) for the test set of 75 docking instances. Each cell contains the median binding energy values (kcal/mol) from 30 independent runs for each molecule (with PDB accession code) and algorithm.

Algs./	PDB	1A94	1A9M	1AAQ	1B6J	1B6K	1B6L	1B6P	1B6M	1BDL	1BDQ
jMetal	gGA	2.72	0.075	−0.83	−2.23	−2.49	−3.915	−3.365	−3.61	−0.965	−1.145
	ssGA	4.1	0.755	0.13	−0.685	−2.285	−2.89	−2.28	−2.435	−0.4	0.375
	DE	0.7	−3.18	−1.865	−3.885	−5.26	−6.04	−6.985	−5.865	−2.275	−3.09
	PSO	5.545	1.785	1.025	−0.43	−0.965	−2.83	−1.6	−1.08	0.755	0.975
ADock	GA	4.35	1.56	0.785	−0.575	−1.33	−3.505	−2.445	−2.49	0.415	0.27
	LGA	1.41	−1.695	−1.8	−3.45	−4.83	−6.27	−5.285	−5.155	−2.36	−2.64

Algs./	PDB	1BDR	1BV7	1BV9	1BWA	1BWB	1D4K	1D4L	1DMP	1G35	1GNM
jMetal	gGA	−1.895	−3.345	−4.105	−2.14	−2.96	−6.33	−6.805	−2.99	−3.21	−3.145
	ssGA	−0.27	−2.18	−2.19	−2.715	−1.955	−4.235	−4.95	−1.885	−2.065	0.85
	DE	−2.83	−6.13	−5.8	−5.47	−5.345	−8.135	−9.45	−5.605	−5.43	−4.4
	PSO	0.315	−1.14	−0.81	−0.7	−0.46	−2.21	−2.52	−1.06	−0.965	2.165
ADock	GA	0.18	−1.715	−2.09	−1.725	−1.69	−4.09	−4.25	−1.725	−1.55	0.665
	LGA	−2.005	−5.035	−4.51	−4.995	−4.765	−6.795	−7.39	−4.235	−4.015	−3.245

Algs./	PDB	1GNN	1GNO	1HBV	1HEF	1HEG	1HIH	1HIV	1HOS	1HPO	1HPS
jMetal	gGA	−3.13	1.495	−1.665	0.805	−1.525	−0.9	0.135	1.85	−3.085	1.42
	ssGA	−0.215	3.52	−0.63	2.035	−0.21	0.555	0.6	1.72	−3.06	1.97
	DE	−5.825	−0.255	−3.71	−1.5	−4.135	−2.505	−1.64	−2.31	−5.305	−3.465
	PSO	1.665	5.98	0.375	2.48	1.185	1.265	2.48	6.26	−2.04	4.57
ADock	GA	0.03	5.22	0.05	1.375	0.215	0.545	1.5	2.245	−3.035	2.025
	LGA	−3.155	0.99	−2.76	−2.225	−3.515	−1.675	−1.645	5.325	−6.075	−2.38

Algs./	PDB	1HPV	1HSG	1HTE	1HTG	1HVI	1HVJ	1HVK	1HVL	1HVS	1HWR
jMetal	gGA	−1.73	−2.9	−2.195	0.41	1.115	1.075	1.13	0.895	0.59	−2.645
	ssGA	−0.915	−1.885	−2.08	1.82	1.65	2.2	2.26	1.925	0.905	−1.685
	DE	−3.635	−5.805	−5.285	−4.83	−2.09	−1.775	−1.555	−1.79	−2.7	−5.03
	PSO	0.125	−0.335	−1.295	6.015	2.565	2.06	2.475	2.63	2.415	−0.91
ADock	GA	−0.49	−1.185	−1.53	4.255	2.045	1.855	2.6	2.48	2.44	−1.655
	LGA	−2.505	−3.445	−3.98	−1.63	−0.31	−0.645	−0.17	−0.545	−0.81	−3.88

Algs./	PDB	1HXW	1IZH	1IZI	1JLD	1K6C	1K6P	1K6T	1K6V	1KZK	1MES
jMetal	gGA	−0.155	1.255	0.065	−0.41	−3.655	−3.945	−4.245	−3.905	−2.755	−2.92
	ssGA	0.295	1.145	1.195	−0.15	−3.16	−2.7	−3.32	−2.57	−1.035	−1.755
	DE	−3.565	−2.005	−1.83	−2.225	−4.85	−6.39	−9.205	−6.165	−5.645	−6.06
	PSO	2.145	2.785	2.255	1.23	−0.925	−1.03	−0.995	−0.965	0.205	−0.545
ADock	GA	1.12	1.98	1.34	1.06	−2.52	−2.16	−2.23	−1.97	−0.99	−1.19
	LGA	−1.66	−0.705	−1.09	−2.03	−5.55	−4.575	−5.435	−5.07	−4.565	−4.045

Algs./	PDB	1MEU	1MTR	1MUI	1ODY	1PRO	1QBR	1QBT	1QBU	1SBG	1TCX
jMetal	gGA	−2.97	−3.995	−1.18	0.045	−2.38	−2.75	−4.325	−2.21	−1.03	−1.52
	ssGA	−2.25	−1.775	0.19	0.725	−1.745	−1.895	−3.13	−2.87	−0.31	−0.455
	DE	−5.74	−5.62	−2.645	−1.95	−4.925	−5.925	−6.055	−6.81	−2.51	−2.605
	PSO	−1.155	−0.86	0.85	2.295	0.175	−0.525	−1.47	−1.23	0.785	0.525
ADock	GA	−2.09	−1.9	0.575	1.35	−1.585	−1.59	−2.615	−1.345	−0.12	−0.205
	LGA	−4.435	−5.065	−1.98	−1.625	−3.855	−4.69	−5.85	−3.755	−2.3	−2.36

Algs./	PDB	1VIJ	1VIK	1Z1H	1Z1R	2BPV	2BPX	3AID	3TLH	4HVP	4PHV
jMetal	gGA	1.21	−0.155	−4.06	−3.97	−2.095	−2.8	−0.265	−0.705	2.365	12.51
	ssGA	2.475	1.975	−4.245	−3.09	−1.045	−0.635	−0.035	−1.09	3.24	13.835
	DE	−1.09	−1.725	−5.16	−5.12	−3.89	−4.575	−3.44	−4.855	0.19	7.28
	PSO	3.88	3.86	−2.81	−2.205	2.44	−0.615	0.71	0.01	4.785	16.905
ADock	GA	2.96	2.98	−4.175	−3.15	−1.195	−1.145	−0.07	−1.39	4.45	15.025
	LGA	−0.64	−0.11	−7.27	−5.655	0.715	−3.595	−2.68	−4.28	1.235	7.94

Algs./	PDB	5HVP	7HVP	7UPJ	8HVP	9HVP
jMetal	gGA	−0.075	−2.29	−3.645	−1.84	−0.175
	ssGA	0.525	−2.105	−2.555	−1.475	0.3
	DE	−2.515	−3.53	−5.47	−4.26	−2.57
	PSO	2.155	−1.075	−1.855	1.79	1.245
ADock	GA	1.26	−1.54	−2.52	−1.385	1.305
	LGA	−2.225	−2.965	−5.155	−0.285	−1.87

established as the control algorithm for the post-hoc Holm test, which is compared with the remaining algorithms. The adjusted p-values (right-hand column in Table 7) resulting from these comparisons are, in all cases, lower than the confidence level ($\alpha = 0.05$), meaning that the DE algorithm of jMetal is statistically better than all the other algorithms studied here. It is worth noting that LGA, with operators and local search methods specifically designed for the molecular docking problem, also shows a lower performance than DE, the latter with (canonical) generic operation. This leads one to suspect that Differential Evolution performs a well-adapted learning procedure to the problem tackled, thus emerging as a useful base-line technique for future proposals that take specific information from docking experiments.

From a graphical point of view, Fig. 3 shows the boxplots of the median binding energy (kcal/mol) distributions of the six compared algorithms, and for the 75

Table 6
Performance results of compared algorithms (gGA, ssGA, DE, and PSO from jMetal; GA and LGA from AutoDock) for the test set of 75 docking instances. Each cell contains the best binding energy value (kcal/mol) from 30 independent runs for each molecule (with PDB accession code) and algorithm.

Algs./	PDB	1A94	1A9M	1AAQ	1B6J	1B6K	1B6L	1B6P	1B6M	1BDL	1BDQ
jMetal	gGA	−0.03	−2.89	−2.23	−4.53	−5.17	−7.81	−6.94	−6.79	−5.7	−4.07
	ssGA	0.71	−2.25	−2.96	−2.69	−5.02	−7.38	−5.33	−5.94	−2.97	−3.39
	DE	−0.33	−3.3	−2.96	−5.74	−6.47	−7.88	−9.68	−8.44	−5.89	−4.99
	PSO	0.36	−0.72	−1.61	−3.48	−3.66	−5.7	−3.89	−5.34	−1.57	−1.96
ADock	GA	1.35	−1.59	−0.96	−3.03	−3.26	−7.16	−4.82	−4.63	−3.05	−2.77
	LGA	−9.23	−2.97	−7.01	−4.82	−7.58	−10.17	−7.72	−11.5	−5.03	−3.8

Algs./	PDB	1BDR	1BV7	1BV9	1BWA	1BWB	1D4K	1D4L	1DMP	1G35	1GNM
jMetal	gGA	−3.31	−5.9	−6.26	−5.45	−6.91	−8.65	−10.41	−5.52	−6.26	−5.15
	ssGA	−3.21	−6.53	−5.86	−4.89	−4.67	−8.74	−8.84	−4.43	−6.1	−5.67
	DE	−4.5	−7.57	−7.65	−7.74	−6.02	−10.86	−11.18	−6.47	−6.21	−5.84
	PSO	−2.82	−5.38	−4.55	−3.49	−5.18	−6.45	−8.09	−4.07	−4.01	−2.72
ADock	GA	−3.6	−3.98	−5.39	−4.59	−4.49	−8.53	−7.92	−4.37	−4.6	−3.07
	LGA	−3.85	−7.29	−6.71	−7.02	−6.18	−11.28	−13.28	−6.84	−6.35	−18.69

Algs./	PDB	1GNN	1GNO	1HBV	1HEF	1HEG	1HIH	1HIV	1HOS	1HPO	1HPS
jMetal	gGA	−7.05	−0.84	−3.09	−2.38	−3.07	−2.78	−2.88	−0.32	−5.03	−1.61
	ssGA	−5.8	0.72	−2.8	−1.67	−3.67	−1.72	−3.4	−1.25	−5.05	−2.54
	DE	−7.75	−2.39	−4.6	−4.33	−5.35	−4.09	−2.83	−4.91	−6.06	−6.48
	PSO	−4.85	2.45	−2.28	−2.85	−0.82	−1.91	−0.53	4.11	−4.96	−2.81
ADock	GA	−2.84	2.67	−2.1	0.12	−2.79	−1.05	−1.76	−0.69	−5.53	−0.38
	LGA	−19.18	−14.71	−4.52	−3.76	−5.89	−3.12	−4.03	3.28	−9.85	−7.49

Algs./	PDB	1HPV	1HSG	1HTE	1HTG	1HVI	1HVJ	1HVK	1HVL	1HVS	1HWR
jMetal	gGA	−3.67	−5.7	−5.57	−3.93	−2.53	−2.26	−1.85	−1.75	−1.97	−5.16
	ssGA	−3.23	−4.68	−4.21	−1.73	−1.25	−1.22	−1.35	−0.88	−1.96	−2.84
	DE	−5.17	−7.1	−6.92	−5.0	−3.06	−2.5	−2.75	−2.37	−2.98	−5.84
	PSO	−2.27	−2.72	−3.02	−3.38	−2.38	−1.03	−0.05	−0.21	−0.26	−2.94
ADock	GA	−2.32	−2.92	−4.74	1.22	0.22	−0.54	0.72	0.79	0.22	−4.23
	LGA	−3.51	−4.98	−5.78	−6.74	−2.29	−2.89	−2.85	−2.23	−3.15	−8.0

Algs./	PDB	1HXW	1IZH	1IZI	1JLD	1K6C	1K6P	1K6T	1K6V	1KZK	1MES
jMetal	gGA	−3.79	−1.86	−2.11	−3.63	−7.62	−6.63	−8.28	−7.07	−6.0	−4.54
	ssGA	−3.19	−3.06	−2.26	−2.77	−6.92	−6.68	−6.43	−6.39	−6.84	−4.26
	DE	−3.98	−2.96	−4.41	−3.65	−9.84	−8.85	−10.24	−8.24	−7.91	−8.93
	PSO	−1.28	0.02	−1.41	−1.63	−5.14	−4.69	−4.85	−3.7	−4.31	−3.36
ADock	GA	−1.01	−0.56	−0.43	−1.19	−5.44	−4.26	−6.58	−4.45	−3.42	−4.3
	LGA	−4.14	−3.15	−4.26	−5.11	−8.64	−6.94	−8.93	−7.72	−7.51	−7.41

Algs./	PDB	1MEU	1MTR	1MUI	1ODY	1PRO	1QBR	1QBT	1QBU	1SBG	1TCX
jMetal	gGA	−5.73	−6.42	−2.78	−2.66	−4.91	−6.63	−6.27	−5.93	−3.79	−3.21
	ssGA	−4.58	−4.29	−3.14	−1.06	−5.22	−5.88	−5.51	−5.47	−2.57	−2.78
	DE	−9.09	−8.1	−4.77	−5.6	−6.69	−6.65	−9.19	−8.41	−4.29	−3.91
	PSO	−5.15	−3.74	−2.91	−1.37	−2.28	−3.86	−5.19	−4.32	−3.48	−1.84
ADock	GA	−4.06	−4.12	−2.13	−0.9	−3.88	−4.54	−5.28	−3.16	−1.51	−1.35
	LGA	−8.88	−6.25	−4.87	−4.33	−5.48	−6.06	−12.86	−8.61	−4.72	−3.8

Algs./	PDB	1VIJ	1VIK	1Z1H	1Z1R	2BPV	2BPX	3AID	3TLH	4HVP	4PHV
jMetal	gGA	−2.72	−2.41	−7.95	−6.99	−4.75	−6.04	−3.22	−3.66	−0.82	9.1
	ssGA	−1.43	−1.6	−10.96	−5.48	−4.11	−4.68	−2.0	−3.74	−0.74	7.23
	DE	−2.92	−2.92	−8.09	−7.4	−4.79	−6.61	−4.13	−7.12	−0.87	7.17
	PSO	0.72	0.02	−7.1	−4.91	1.69	−2.73	−1.62	−2.15	1.99	8.67
ADock	GA	−0.64	0.54	−7.34	−4.7	−3.66	−3.79	−1.98	−3.6	0.9	11.85
	LGA	−10.17	−2.27	−9.38	−8.67	−1.37	−5.85	−5.37	−7.14	0.21	−9.92

Algs./	PDB	5HVP	7HVP	7UPJ	8HVP	9HVP
jMetal	gGA	−2.1	−3.68	−5.3	−4.26	−2.9
	ssGA	−2.09	−3.86	−6.69	−4.36	−2.41
	DE	−4.42	−4.53	−7.37	−5.44	−5.75
	PSO	−1.09	−5.81	−4.87	0.3	−1.81
ADock	GA	−1.89	−2.92	−3.98	−4.44	−3.12
	LGA	−9.01	−3.94	−8.39	−2.38	−3.4

molecular docking instances. This figure provides a general overview of results, where it is clearly observable how DE obtains the best distribution of binding energies, for the instances considered. Regarding the other techniques, as ranked in Table 7, the boxplot reveals that the two GA versions of jMetal: ssGA and gGA, outperform the GA version of AutoDock. Between the gGA and ssGA, the former appears to be the most effective.

The lowest ranked algorithm is PSO which, in spite of its traditionally successful performance in multiple continuous optimization problems [59], does not perform

Table 7
Average Friedman's rankings with Holm's Adjusted p-values ($\alpha = 0.05$) of compared algorithms (gGA, ssGA, DE, and PSO from jMetal; GA and LGA from AutoDock) for the test set of 75 docking instances. Symbol * indicates the Control algorithm.

Algorithm	Friedman's rank	Holm's adjusted $_{p-\text{value}}$
*DE	1.10	–
LGA	2.02	$2.45E-03$
gGA	3.05	$4.25E-11$
ssGA	4.07	$1.31E-25$
GA	4.78	$1.19E-37$
PSO	5.94	$7.20E-62$

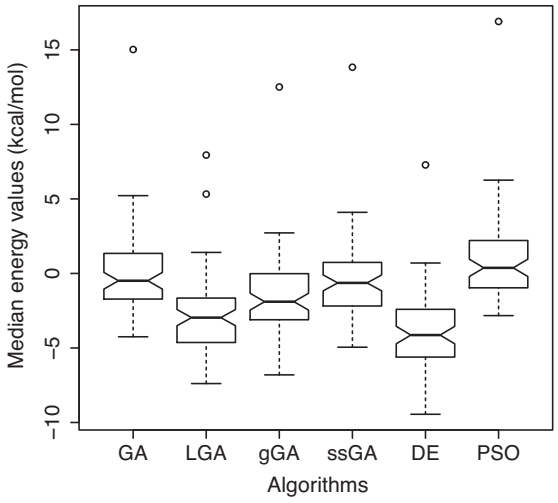

Fig. 3. Boxplots of median binding energy (kcal/mol) distributions of the six compared algorithms, and for the 75 molecular docking instances. It is clearly observable that DE obtains the best distribution of results, for the worked instances.

efficiently on the set of molecular instances in this study. Probably, the use of other formulated versions of PSO together with a more exhaustive parameter setting, would lead into a better performance of this metaheuristic.

A last observation on numerical performance concerns the internal behavior of jMetal algorithms. Fig. 4 plots the evolution traces (fitness) of the best individuals found so far, through the iteration process of evaluated algorithms, for molecules 1BV7 and 1HWR. These two instances have been selected as they represent heterogeneous molecules in cyclic urea (1BV7) and large size (1HWR) groups, although with similar resolution (Å) ranges. In the two plots, a clear observation involves the evolution traces of DE, that show a later convergence with regards to the other algorithms, although it reaches lower fitness values just before the midpoint of the evolution progress. This behavior sheds light on the balanced trade-off between

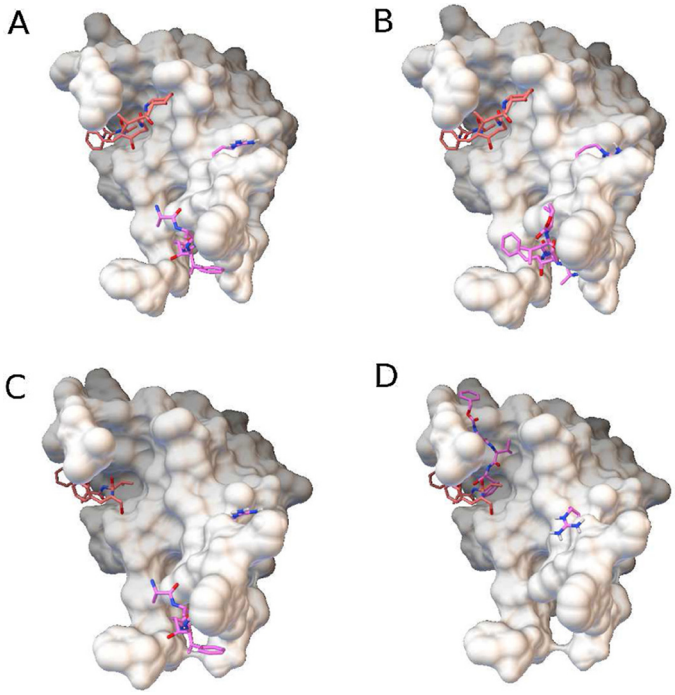

Fig. 5. Best energy conformations for LGA (left molecules) and DE (right molecules) algorithms in the 1HEG (molecules at the top) and 3TLH (molecules at the bottom) instances.

exploration-exploitation [60] that the DE performs when dealing with different molecular docking instances. Nevertheless, in the case of gGA, it performs a fast convergence behavior also with successful solutions. In concrete, for 1HWR molecule (and similar for 1BV7), gGA stagnates close to 250,000 evaluations, although showing similar results to DE. This result suggests that generational GA of jMetal could be also a good choice when looking for fast, but good enough solutions.

7.2. Biological analysis

For the qualitative analysis, we focused on RMSD median results obtained from the DE and LGA algorithms (see Table 8). According to the data shown, the lowest RMSD scores were obtained by the DE algorithm in comparison with the LGA. In fact, taking into account the classification of the inhibitor size shown in Table 2, the DE algorithm has solutions with lower RMSD values than the LGA ones for large ligands (11), small ligands (14), medium ligands (12) and urea cyclic ligands (7). To the contrary, the results obtained from the LGA algorithm were lower for large ligands (7), small ligands (9), medium ligands (10) and urea cyclic ligands (6). Therefore, DE yields a lower RMSD score average than the LGA.

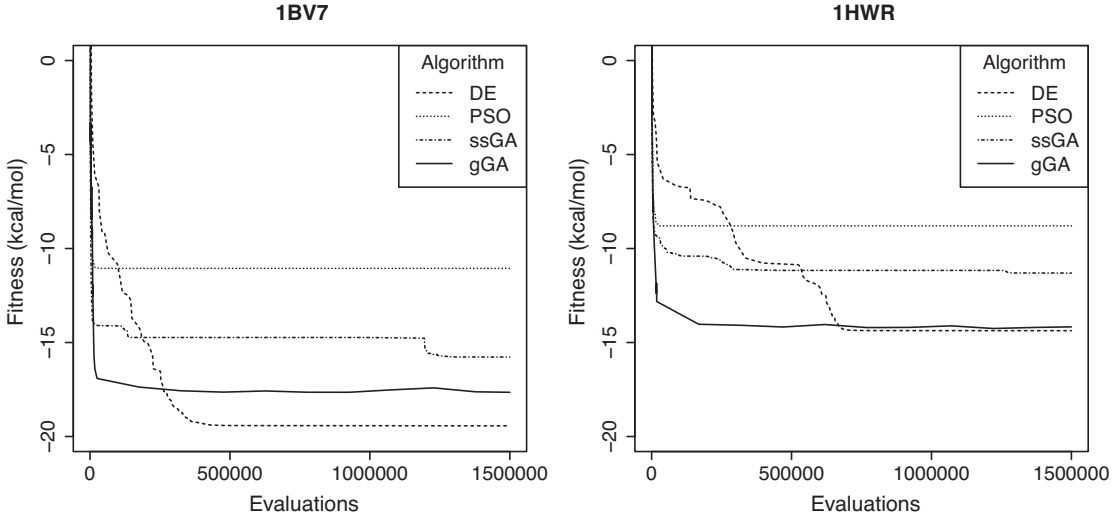

Fig. 4. Fitness evolution for the jMetal algorithms in the 1BV7 and 1HWR instances.

Table 8
Performance results of compared algorithms (gGA, ssGA, DE, and PSO from jMetal; GA and LGA from AutoDock) for the test set of 75 docking instances. Each cell contains the median RMSD value out of 30 independent runs for each molecule (with PDB accession code) and algorithm.

Algs./	PDB	1A94	1A9M	1AAQ	1B6J	1B6K	1B6L	1B6P	1B6M	1BDL	1BDQ
jMetal	gGA	12.79	11.11	13.11	13.48	13.26	12.48	12.92	12.78	10.29	12.56
	ssGA	13.57	12.57	13.32	13.33	13.91	12.90	13.16	13.42	10.52	13.68
	DE	12.51	9.56	12.78	11.34	11.38	12.76	12.09	12.26	11.31	11.36
	PSO	13.99	12.93	12.91	13.12	13.18	13.67	12.61	12.75	12.28	13.50
ADock	GA	13.26	13.05	14.90	13.41	13.34	14.28	14.18	13.16	12.26	14.00
	LGA	12.39	11.98	12.70	14.20	10.14	13.04	12.54	11.88	10.54	13.11

Algs./	PDB	1BDR	1BV7	1BV9	1BWA	1BWB	1D4K	1D4L	1DMP	1G35	1GNM
jMetal	gGA	13.43	11.6	13.28	11.78	10.98	12.69	12.50	12.70	13.00	12.47
	ssGA	13.35	13.16	12.98	12.04	12.70	12.80	12.76	13.28	13.00	13.45
	DE	10.68	12.96	13.04	12.96	12.92	13.10	12.79	10.48	12.51	10.38
	PSO	13.39	13.50	12.20	11.84	13.18	12.28	12.20	13.06	12.28	12.68
ADock	GA	14.22	12.66	12.12	11.30	12.68	13.14	13.42	14.28	13.42	12.72
	LGA	12.30	12.23	13.50	11.58	12.45	11.91	11.12	11.78	12.53	10.96

Algs./	PDB	1GNN	1GNO	1HBV	1HEF	1HEG	1HIH	1HIV	1HOS	1HPO	1HPS
jMetal	gGA	12.37	11.95	12.48	9.84	10.02	13.58	12.27	8.61	10.98	12.34
	ssGA	12.32	12.65	12.30	9.61	10.25	14.60	12.45	11.65	12.57	11.58
	DE	12.36	13.19	13.70	11.34	15.93	12.32	12.76	13.36	9.66	10.33
	PSO	12.54	13.43	12.15	10.59	10.58	13.54	13.17	9.28	13.92	12.18
ADock	GA	11.64	12.30	14.42	10.48	11.05	15.14	13.64	10.12	12.82	10.04
	LGA	12.04	11.42	12.66	7.63	15.41	13.68	11.60	9.30	8.51	10.66

Algs./	PDB	1HPV	1HSG	1HTE	1HTG	1HVI	1HVJ	1HVK	1HVL	1HVS	1HWR
jMetal	gGA	12.37	12.62	12.32	12.28	13.14	13.36	13.72	12.83	13.53	12.77
	ssGA	13.45	12.80	12.76	11.96	12.88	13.12	11.64	13.47	13.32	13.33
	DE	12.74	11.99	16.19	12.04	11.12	12.72	12.00	11.76	11.02	10.32
	PSO	12.16	12.92	14.54	11.79	13.32	14.05	12.20	13.44	13.42	14.16
ADock	GA	14.54	12.89	13.16	12.38	12.98	13.18	13.07	13.91	13.25	13.58
	LGA	14.65	12.12	13.48	13.23	13.52	13.18	12.36	12.96	12.52	13.07

Algs./	PDB	1HXW	1IZH	1IZI	1JLD	1K6C	1K6P	1K6T	1K6V	1KZK	1MES
jMetal	gGA	11.40	13.02	10.75	12.58	13.30	12.96	12.39	12.81	12.22	11.88
	ssGA	12.56	12.73	12.34	13.72	12.86	13.16	13.37	12.55	12.86	13.75
	DE	12.35	12.25	9.49	12.15	13.46	12.60	11.94	13.09	11.74	10.98
	PSO	13.36	13.36	13.89	13.04	13.37	13.52	13.30	13.17	13.66	13.96
ADock	GA	13.80	14.21	12.37	13.66	13.29	13.37	12.93	13.13	13.18	12.88
	LGA	12.60	11.74	11.75	12.14	12.67	13.71	12.80	12.52	12.25	11.76

Algs./	PDB	1MEU	1MTR	1MUI	1ODY	1PRO	1QBR	1QBT	1QBU	1SBG	1TCX
jMetal	gGA	12.50	12.81	12.53	11.81	11.20	11.86	12.48	10.84	12.48	12.75
	ssGA	13.17	13.68	12.20	12.57	12.88	11.84	12.43	12.12	14.16	13.39
	DE	12.16	12.57	11.12	12.67	11.89	13.13	12.33	11.33	11.20	12.25
	PSO	12.61	13.28	12.48	13.58	13.72	12.41	12.95	13.32	13.48	13.94
ADock	GA	13.61	13.87	14.00	11.02	13.06	12.75	11.46	13.40	14.86	14.38
	LGA	12.62	12.14	12.90	11.88	11.84	11.84	11.62	12.52	12.86	12.42

Algs./	PDB	1VIJ	1VIK	1Z1H	1Z1R	2BPV	2BPX	3AID	3TLH	4HVP	4PHV
jMetal	gGA	12.85	12.44	13.02	12.52	11.42	12.42	14.08	9.98	13.36	13.20
	ssGA	13.00	12.32	12.94	13.04	13.72	13.04	14.25	9.83	13.00	12.30
	DE	10.61	12.36	11.36	11.49	10.58	12.64	11.84	2.92	11.32	11.83
	PSO	12.78	12.83	14.57	13.02	12.64	13.72	13.77	11.15	13.39	12.86
ADock	GA	12.09	14.12	13.54	13.26	11.56	13.17	13.92	11.19	13.64	13.04
	LGA	11.99	12.92	13.20	12.66	11.88	12.96	13.09	6.23	12.55	11.83

Algs./	PDB	5HVP	7HVP	7UPJ	8HVP	9HVP
jMetal	gGA	13.12	14.44	12.84	14.84	11.32
	ssGA	12.68	13.74	13.56	13.96	12.21
	DE	13.06	15.11	9.88	15.79	11.07
	PSO	13.10	15.54	13.11	15.12	12.30
ADock	GA	13.00	15.25	13.00	13.20	11.40
	LGA	11.80	17.12	12.66	13.54	9.54

Two instances 1HEG and 3TLH have been selected from large and small ligand groups to visually analyze how the compounds are bonded of with respect to the reference ligand. Therefore, Fig. 5 shows the HIV-protease receptors (surface shaded in gray), the computational docked (purple color) and the reference ligands (red color). The top left (A) and right (B) images correspond with the 1HEG conformations with the best energy score returned by the LGA and DE, respectively. The bottom left (C) and right (D) images are those of 3TLH conformations with the best energy returned by the LGA and DE, respectively.

As shown in A and B, the ligand pose predicted by the LGA is located far from the reference ligand binding site, this being a solution with a lower RMSD value than the one returned by DE. Although the ligand position is also far from the reference ligand, the computed ligand position is closer. In this context, we observe

that, although the DE algorithm yields lower RMSD score results than LGA, most of the computed ligand poses were still located far from the reference ligand binding site. These results are in accordance with the ones obtained by Morris et al. [3] when the flexibility was added to the receptor and the results obtained were worse for small molecules. These results could be explained due to the search space increases with the receptor flexibility and then, the size of the ligand makes difficult to find the correct conformations. Furthermore, as images C and D show, the DE docked ligand finds a closer position to the reference ligand, and therefore the LGA computed ligand has a worse position. These results are explained due to the fact that large ligands are actually easier problems than small ones. This is because the large ligands are forcibly bonded to the binding site according to their size.

8. Conclusions and future work

The main objective of this work has been to carry out a comparative study using metaheuristics from the jMetal framework and AutoDock 4.2. The selected algorithms were two GAs, a PSO, and a DE, provided by jMetal. To achieve this goal, we have used the C++ version of jMetal that has been integrated with AutoDock, resulting in a software tool which allows the metaheuristics included in jMetal to be used easily.

An extensive set of PDB structures has been selected to compare the search methods involving flexibility in HIV-protease macromolecules and ligands, as this result in a more realistic and complex problem. Ligands in all instances present a size range from small to large, including cyclic urea.

The main conclusions to be drawn are listed as follows:

- In general, the DE (jMetal) optimizer shows the best performance results, even with statistical confidence in comparison with the other evaluated metaheuristics. DE also outperforms well-known algorithms techniques in the state of the art, LGA and GA (AutoDock), even though they have operators designed specifically for the problem domain.
- DE converges later, although to high quality solutions. This behavior of DE implies a balanced trade-off between exploration-exploitation when dealing with different molecular docking instances.
- In the case of gGA, it demonstrates a fast convergence behavior also with successful solutions. Specifically, for the 1HWR molecule (and similar for 1BV7), gGA stagnates close to 250,000 evaluations, although showing similar results to DE. This result suggests that generational GA of jMetal could be also a good choice when looking for fast, but good enough solutions.
- jMetal-AutoDock is shown to be a useful tool for three kinds of researchers: first, those interested in efficient docking search methods that can be applied in the drug screening domain; second, metaheuristic designers that can use protein docking as a real-world case study and third, those biological researchers that are focused on the problem of macromolecule flexibility such as Abreu Rui et al. that improve the docking scores through a study of selective flexibility of the side-chain residues of VEGFR-2 tyrosine kinase receptor [61].

A number of future lines of research emerge from this paper. The jMetal was originally designed to deal with multi-objective optimization problems and incorporates many metaheuristics, so the next natural step is to carry out a study by using a multi-objective formulations of the docking problem. Furthermore, bearing in mind that DE does not apply any knowledge of the problem being solved, an open research line deserving of study is whether its search capabilities can be improved by using a local search and any problem knowledge. This could also be applicable to PSO, ssGA and gGA. Finally, AutoDock Vina is considered as a new generation of docking techniques, so future work could involve integrating the jMetal framework inside it.

Acknowledgment

This work is partially funded by Grants TIN2014-58304-R (Ministerio de Economía y Competitividad), TIN2011-25840 (Ministerio de Ciencia e Innovación), P11-TIC-7529 and P12-TIC-1519 (Plan Andaluz de Investigación, Desarrollo e Innovación).

References

[1] I.D. Kuntz, J.M. Blaney, S.J. Oatley, R. Langridge, T.E. Ferrin, A geometric approach to macromolecule–ligand interactions, J. Mol. Biol. 161 (2) (1982) 269–288.
[2] F.C. Bernstein, T.F. Koetzle, G.J. Williams, E.F. Meyer, M.D. Brice, J.R. Rodgers, O. Kennard, T. Shimanouchi, M. Tasumi, The Protein Data Bank: a computer-based archival file for macromolecular structures, J. Mol. Biol. 112 (3) (1977) 535–542.
[3] G.M. Morris, R. Huey, W. Lindstrom, M.F. Sanner, R.K. Belew, D.S. Goodsell, A.J. Olson, AutoDock4 and AutoDockTools4: automated docking with selective receptor flexibility, J. Comput. Chem. 30 (16) (2009) 2785–2791.
[4] C. Blum, J. Puchinger, G.R. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: a survey, Appl. Soft Comput. 11 (6) (2011) 4135–4151, http://dx.doi.org/10.1016/j.asoc.2011.02.032.
[5] J. Kennedy, R. Eberhart, Particle swarm optimization, in: IEEE IJCNN, Vol. 4, 1995, pp. 1942–1948.
[6] M. Dorigo, T. Stützle, Ant Colony Optimization, Bradford Company, Scituate, MA, USA, 2004.
[7] S.F. Sousa, P.A. Fernandes, M.J. Ramos, Protein ligand docking: current status and future challenges, Proteins 65 (1) (2006) 15–26.
[8] G.M. Morris, D.S. Goodsell, R.S. Halliday, R. Huey, W.E. Hart, R.K. Belew, A.J. Olson, Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function, J. Comput. Chem. 19 (1998) 1639–1662.
[9] R. Storn, K. Price, Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[10] J.J. Durillo, A.J. Nebro, jmetal: a java framework for multi-objective optimization, Adv. Eng. Softw. 42 (10) (2011) 760–771.
[11] E. López-Camacho, M.J. García-Godoy, A.J. Nebro, J.F.A. Montes, jMetal-Cpp: optimizing molecular docking problems with a C++ metaheuristic framework, Bioinformatics 30 (3) (2014) 437–438, http://dx.doi.org/10.1093/bioinformatics/btt679.
[12] M. Crepinsek, S.-H. Liu, M. Mernik, Replication and comparison of computational experiments in applied evolutionary computing: common pitfalls and guidelines to avoid them, Appl. Soft Comput. 19 (2014) 161–170.
[13] E.-W. Lameijer, T. Bäck, J.N. Kok, A.P. Ijzerman, Evolutionary algorithms in drug design, Nat. Comp. 4 (3) (2005) 177–243.
[14] S. Cosconati, S. Forli, A.L. Perryman, R. Harris, D.S. Goodsell, A.J. Olson, Virtual screening with AutoDock: theory and practice, Expert Opin. Drug Discov. 5 (6) (2010) 597–607.
[15] D.S. Goodsell, A.J. Olson, Automated docking of substrates to proteins by simulated annealing, Proteins 8 (1990) 195–202.
[16] O. Trott, A.J. Olson, Autodock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading, J. Comput. Chem. 31 (2) (2010) 455–461.
[17] E. Atilgan, J. Hu, Efficient protein–ligand docking using sustainable evolutionary algorithm, in: GECCO, 2010, pp. 211–212.
[18] H.-M. Chen, B.-F. Liu, H.-L. Huang, S.-F. Hwang, S.-Y. Ho, Sodock: swarm optimization for highly flexible protein–ligand docking, J. Comput. Chem. 28 (2) (2007) 612–623.
[19] V. Namasivayam, R. Günther, Research article: pso@autodock: a fast flexible molecular docking program based on swarm intelligence, Chem. Biol. Drug Des. 70 (6) (2007) 475–484.
[20] Y. Liu, L. Zhao, W. Li, D. Zhao, M. Song, Y. Yang, Fipsdock, A new molecular docking technique driven by fully informed swarm optimization algorithm, J. Comput. Chem. 34 (2012) 67–75.
[21] R. Thomsen, Flexible ligand docking using evolutionary algorithms: investigating the effects of variation operators and local search hybrids, Biosystems 72 (2) (2003) 57–73.
[22] M. Koohi-Moghadam, A.T. Rahmani, Molecular docking with opposition-based differential evolution, in: Proceedings of the 27th Annual ACM Symposium on Applied Computing SAC '12, ACM, 2012, pp. 1387–1392.
[23] H.W. Chung, S.J. Cho, K.-R. Lee, K.-H. Lee, Self-adaptive differential evolution algorithm incorporating local search for protein–ligand docking, J. Phys.: Conf. Ser. 410 (1) (2013) 012030.
[24] R. Meier, M. Pippel, F. Brandt, W. Sippl, C. Baldauf, ParaDockS: a framework for molecular docking with population-based metaheuristics, J. Chem. Inf. Model. 50 (5) (2010) 879–889.
[25] O. Korb, T. Stutzle, T.E. Exner, Application of ant colony optimization to structure-based drug design, in: M. Dorigo, et al. (Eds.), Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2006, Ser. Technical Report Series: TR/IRIDIA/2006-023 11 LNCS, Springer Verlag, 2006, pp. 247–258.
[26] M. Simonsen, C.N. Pedersen, M.H. Christensen, R. Thomsen, GPU-accelerated high-accuracy molecular docking using guided differential evolution: real world applications, in: Proceedings of the 13th annual conference on Genetic

and evolutionary computation, GECCO '11, ACM, New York, NY, USA, 2011, pp. 1803–1810.
[27] L. Kang, X. Wang, Multi-scale optimization model and algorithm for computer-aided molecular docking, in: 2012 Eighth International Conference on Natural Computation (ICNC), 2012, pp. 1208–1211.
[28] G. Heberlé, W. de Azevedo Jr., Bio-inspired algorithms applied to molecular docking simulations, Curr. Med. Chem. 18 (9) (2011) 1339–1352.
[29] J. Tavares, A.-A. Tantar, N. Melab, E.-G. Talbi, The influence of mutation on protein–ligand docking optimization: a locality analysis, in: Proceedings of the 10th international conference on Parallel Problem Solving from Nature: PPSN X, Springer-Verlag, 2008, pp. 589–598.
[30] C.A. Nicolaou, N. Brown, C.S. Pattichis, Molecular optimization using computational multi-objective methods, Curr. Opin. Drug Disc. Dev. 10 (3) (2007) 316–324.
[31] O. Nicolotti, I. Giangreco, A. Introcaso, F. Leonetti, A. Stefanachi, A. Carotti, Strategies of multi-objective optimization in drug discovery and development, Expert Opin. Drug Discov. 6 (9) (2011) 871–884.
[32] A. Grosdidier, V. Zoete, O. Michielin, Eadock: docking of small molecules into protein active sites with a multiobjective evolutionary optimization, Proteins 67 (4) (2007) 1010–1025.
[33] A. Sandoval-Perez, D. Becerra, D. Vanegas, D. Restrepo-Montoya, F. Niño, A multi-objective optimization energy approach to predict the ligand conformation in a docking process, in: EuroGP, 2013, pp. 181–192.
[34] A. Oduguwa, A. Tiwari, S. Fiorentino, R. Roy, Multi-objective optimisation of the protein–ligand docking problem in drug discovery, in: Proceedings of the 8th annual conference on Genetic and evolutionary computation, 2006, pp. 1793–1800.
[35] S. Janson, D. Merkle, M. Middendorf, Molecular docking with multi-objective particle swarm optimization, Appl. Soft Comput. 8 (1) (2008) 666–675.
[36] A version 4.2. Autodock version 4.2 (online).
[37] S.J. Weiner, P.A. Kollman, D.A. Case, U.C. Singh, C. Ghio, G. Alagona, S. Profeta, P. Weiner, A new force field for molecular mechanical simulation of nucleic acids and proteins, J. Am. Chem. Soc. 106 (3) (1984) 765–784.
[38] G.M. Verkhivker, P.A. Rejto, D. Bouzida, S. Arthurs, A.B. Colson, S.T. Freer, D.K. Gehlhaar, V. Larson, B.A. Luty, T. Marrone, P.W. Rose, Towards understanding the mechanisms of molecular recognition by computer simulations of ligand protein interactions, J. Mol. Recognit. 12 (6) (1999) 371–389, doi:10.1002/(SICI)1099-1352(199911/12)12:6< 371::AID-JMR479> 3.0.CO;2-O.
[39] F. Herrera, M. Lozano, J. Verdegay, Tackling real-coded genetic algorithms: operators and tools for behavioural analysis, Artif. Intell. Rev. 12 (4) (1998) 265–319, http://dx.doi.org/10.1023/A:1006504901164.
[40] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308.
[41] jMetalCpp Framework, jmetalcpp framework, 2012, URL: http://sourceforge.net/projects/jmetalcpp
[42] K. Deb, D. Kalyanmoy, Multi-objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, Inc, New York, NY, USA, 2001.
[43] F.J. Solis, R.J.B. Wets, Minimization by random search techniques, Math. Oper. Res. 6 (1) (1981) 19–30.
[44] M. Clerc, J. Kennedy, The particle swarm – explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (1) (2002) 58–73.
[45] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne, The protein data bank, Nucleic Acids Res. 28 (1) (2000) 235–242, http://dx.doi.org/10.1093/nar/28.1.235, arXiv: http://nar.oxfordjournals.org/content/28/1/235.full.pdf+html. URL: http://nar.oxfordjournals.org/content/28/1/235.abstract
[46] K. Backbro, S. Lowgren, K. Osterlund, J. Atepo, T. Unge, J. Hulten, N.M. Bonham, W. Schaal, A. Karlen, A. Hallberg, Unexpected binding mode of a cyclic sulfamide HIV-1 protease inhibitor, J. Med. Chem. 40 (6) (1997) 898–902.
[47] J.D. Tyndall, R.C. Reid, D.P. Tyssen, D.K. Jardine, B. Todd, M. Passmore, D.R. March, L.K. Pattenden, D.A. Bergman, D. Alewood, S.H. Hu, P.F. Alewood, C.J. Birch, J.L. Martin, D.P. Fairlie, Synthesis, stability, antiviral activity, and protease-bound structures of substrate-mimicking constrained macrocyclic inhibitors of HIV-1 protease., J. Med. Chem. 43 (19) (2000) 3495–3504.
[48] W. Schaal, A. Karlsson, G. Ahlsén, J. Lindberg, H.O. Andersson, U.H. Danielson, B. Classon, T. Unge, B. Samuelsson, J. Hultén, A. Hallberg, A. Karlén, Synthesis and comparative molecular field analysis (COMFA) of symmetric and nonsymmetric cyclic sulfamide HIV-1 protease inhibitors, J. Med. Chem. 44 (2) (2001) 155–169.
[49] N. Thanki, J.K. Rao, S.I. Foundling, W.J. Howe, J.B. Moon, J.O. Hui, A.G. Tomasselli, R.L. Heinrikson, S. Thaisrivongs, A. Wlodawer, Crystal structure of a complex of HIV-1 protease with a dihydroxyethylene-containing inhibitor: comparisons with molecular modeling., Protein Sci. 1 (8) (1992) 1061–1072.
[50] E.T. Baldwin, T.N. Bhat, S. Gulnik, B. Liu, I.A. Topol, Y. Kiso, T. Mimoto, H. Mitsuya, J.W. Erickson, Structure of HIV-1 protease with KNI-272, a tight-binding transition-state analog containing allophenylnorstatine, Structure (London, England: 1993) 3 (6) (1995) 581–590.
[51] H. Jhoti, O.M. Singh, M.P. Weir, R. Cooke, P. Murray-Rust, A. Wonacott, X-ray crystallographic studies of a series of penicillin-derived asymmetric inhibitors of HIV-1 protease, Biochemistry 33 (28) (1994) 8417–8427.
[52] P.K. Jadhav, F.J. Woerner, P.Y. Lam, C.N. Hodge, C.J. Eyermann, H.W. Man, W.F. Daneker, L.T. Bacheler, M.M. Rayner, J.L. Meek, S. Erickson-Viitanen, D.A. Jackson, J.C. Calabrese, M. Schadt, C.H. Chang, Nonpeptide cyclic cyanoguanidines as HIV-1 protease inhibitors: synthesis, structure–activity relationships, and X-ray crystal structure studies, J. Med. Chem. 41 (9) (1998) 1446–1455.
[53] M. Mizuno, T. Tonozuka, A. Uechi, A. Ohtaki, K. Ichikawa, S. Kamitori, A. Nishikawa, Y. Sakano, The crystal structure of *Thermoactinomycesvulgaris* R-47 alpha-amylase II (TVA II) complexed with transglycosylated product, Eur. J. Biochem. 271 (12) (2004) 2530–2538.
[54] S. Thaisrivongs, K.D. Watenpaugh, W.J. Howe, P.K. Tomich, L.A. Dolak, K.T. Chong, C.C. Tomich, A.G. Tomasselli, S.R. Turner, J.W. Strohbach, Structure-based design of novel HIV protease inhibitors: carboxamide-containing 4-hydroxycoumarins and 4-hydroxy-2-pyrones as potent nonpeptidic inhibitors, J. Med. Chem. 38 (18) (1995) 3624–3637.
[55] E.F. Pettersen, T.D. Goddard, C.C. Huang, G.S. Couch, D.M. Greenblatt, E.C. Meng, T.E. Ferrin, UCSF chimera – a visualization system for exploratory research and analysis, J. Comput. Chem. 25 (13) (2004) 1605–1612.
[56] D. Thain, T. Tannenbaum, M. Livny, Distributed computing in practice: the condor experience, Concurr. – Pract. Exp. 17 (2-4) (2005) 323–356.
[57] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, Chapman & Hall/CRC, Boca Ratón, Florida, 2007.
[58] A. Norgan, P. Coffman, J.P. Kocher, D. Katzmann, C. Sosa, Multilevel parallelization of AutoDock 4.2, J. Cheminformatics 3 (1) (2011) 12.
[59] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intell. 1 (1) (2007) 33–57, http://dx.doi.org/10.1007/s11721-007-0002-0.
[60] M. Crepinsek, S.-H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: a survey, ACM Comput. Surv. 45 (3) (2013) 35:1–35:33.
[61] R.M.V. Abreu, H.J.C. Froufe, M.J.R.P. Queiroz, I.C.F.R. Ferreira, Selective flexibility of side-chain residues improves VEGFR-2 docking score using Autodock Vina, Chem. Biol. Drug Des. 79 (4) (2012) 530–534.