The Journal of Supercomputing (2023) 79:9538–9557
https://doi.org/10.1007/s11227-022-05014-0

Improving drug discovery through parallelism

Jerónimo S. García[1] · Savíns Puertas-Martín[1,2] · Juana L. Redondo[1] · Juan José Moreno[1] · Pilar M. Ortigosa[1]

Accepted: 14 December 2022 / Published online: 16 January 2023

Abstract
Compound identification in ligand-based virtual screening is limited by two key issues: the quality and the time needed to obtain predictions. In this sense, we designed OptiPharm, an algorithm that obtained excellent results in improving the sequential methods in the literature. In this work, we go a step further and propose its parallelization. Specifically, we propose a two-layer parallelization. Firstly, an automation of the molecule distribution process between the available nodes in a cluster, and secondly, a parallelization of the internal methods (initialization, reproduction, selection and optimization). This new software, called pOptiPharm, aims to improve the quality of predictions and reduce experimentation time. As the results show, the performance of the proposed methods is good. It can find better solutions than the sequential OptiPharm, all while reducing its computation time almost proportionally to the number of processing units considered.

Keywords High-performance computing · Virtual screening · Shape similarity · Evolutionary algorithms

1 Introduction

The emergence of COVID-19, a severe and highly contagious viral disease, has brought the drug development process to the forefront of the world stage [1]. Due to the impact on society, the number of people who have worked in both the public and private sectors has made it possible to obtain a vaccine that considerably reduces health risks in less than a year [2]. However, this is an isolated milestone,

Savíns Puertas-Martín, Juana L. Redondo, Juan José Moreno and Pilar M. Ortigosa have contributed equally to this work.

✉ Juana L. Redondo
jlredondo@ual.es

Extended author information available on the last page of the article

and the reality is that the process usually takes between 12 and 15 years and could cost almost $1 billion [3, 4].

This is a costly and time-consuming process because different stages are involved, from the target identification to the approval of a medicine. One of these initial stages consists of identifying candidate hits. One of the tools involved in this process is High Throughput Screening (HTS) [5] with excellent results [6, 7]. But without a doubt, Virtual Screening (VS) is the reference in this process due to its capacity to find possible molecules reducing time and cost enormously due to the simulation by means of mathematical models [8].

VS methods fall into two categories: Structure-Based (SBVS) [9] and Ligand-Based VS (LBVS) [10]. Depending on the information obtained from the molecules, one method or the other can be applied. In the case of SBVS, it is necessary to know the three-dimensional structure of the therapeutic target, which can be obtained by building molecular models or experimental methods such as Nuclear Magnetic Resonance (NMR) [11] or X-ray crystallography [12]. An example of SBVS is docking, whose aim is to find the best docking position between a ligand and a receptor [13]. The second major category is LBVS and is applied when no structure information is available, so active and inactive ligand information is exploited as well as descriptors comparison. This includes Quantitative Structure-Activity Relationship (QSAR), similarity search techniques using 2D/3D descriptors, and shape matching techniques (global or partial shape comparison between molecules).

One of the main problems faced by virtual screening is the huge databases to be processed, which involves a lot of computational time. Consequently, speeding up this process is of paramount importance in this area. In that sense, different pieces of software have been developed in recent years. The latest of them and designed by us, is OptiPharm, an evolutionary and parameterizable algorithm. OptiPharm has been compared against different state-of-art, getting betters results [14, 15]. The way it was designed allows adapting to the user's needs. In the original work, two configurations valid for any problem were proposed. One called FAST, with few evaluations to obtain results quickly, and another, called ROBUST, where was allowed to explore deeper the search space to find better solutions in a longer but still measured time.

In this paper, we argue that the quality of the solution should not be compromised by the need to find a quick answer. On the contrary, using high-performance computing, we could find alternatives to speed up the process without sacrificing accuracy. The aim of this article is twofold. On the one hand, we investigate whether there is still room for improvement. In other words, we analyze whether we can configure OptiPharm to be more accurate, without worrying about the computing time. As will be seen, OptiPharm can provide much better results if more demanding configurations are considered. On the other hand, we accelerated the virtual screening process by developing a parallel approach based on two levels. The first is related to the number of molecules in the database, while the second concerns the parallelization of OptiPharm.

The rest of the paper is divided as follows. Section 2 describes some important details about OptiPharm, and explains the parameterizable features of the algorithm. Section 3 explains pOptiPharm, the parallel approach. There, many details about the

management of the solutions in the optimization process are given. Section 4 summarizes the experimental context to test the algorithms, as the scoring function, the database, the algorithm configuration and the hardware where the experiments have been carried out. Finally, Sect. 5 shows the obtained results and Sect. 6 summarizes the conclusions and future research, highlighting the major contributions and some remarks.

2 OptiPharm: the sequential algorithm

This section is dedicated to briefly explaining the features of OptiPharm, focusing mainly on those that can influence the design and performance of the parallel version. Algorithm 1 outlines its main procedures. However, we encourage the reader to visit the original article for a better understanding of the algorithm [14].

Algorithm 1 Simplified OptiPharm's algorithm

Require: Two molecules (a query and a target), $N, M, t_{max}, R_{t_{max}}$
▷ Each method in [1-7] updates the counter *globalFuncEval*
1: Initial poses creation (M)
2: Optimization (N)
3: **while** $k < t_{max}$ **do**
4: New poses creation (N)
5: Elitist selection (M)
6: Improvement (N, M)
7: **end while**

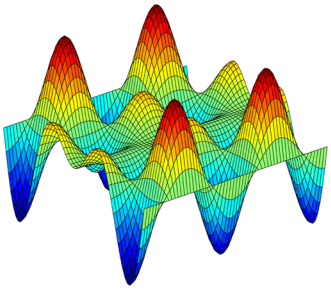
OptiPharm is an evolutionary global optimization algorithm specifically designed to solve similarity maximization problems between two compounds: a query molecule and a target molecule. The latter is rotated and translated to obtain the maximum similarity while the query keeps fixed. To do it, ten decision variables are used: seven of them are used to define a rotation, six to define the two 3D points, and one for the angle; the remaining three are used for the displacement in the three dimensions. This set ϕ of decision variables is known as a solution (individual, pose or point) in OptiPharm.

OptiPharm applies mechanisms based on evolution, basically reproduction-replacement-improvement sequences, to direct a set of initial points toward the global and local optima. Each particular solution has the potential to become a local or a global optimum independently of the remaining ones, that is, a point has the ability to evolve toward the optima without the participation of the remaining ones. This is significant from a parallelism point of view, as intrinsic parallelism can be easily exploited by dividing the number of candidate solutions by the number of available processing units. To this end, each candidate solution has a maximum budget in the number of function evaluations (f.e.) to reach an optimum, although not all of them are always consumed. This is because OptiPharm procedures are

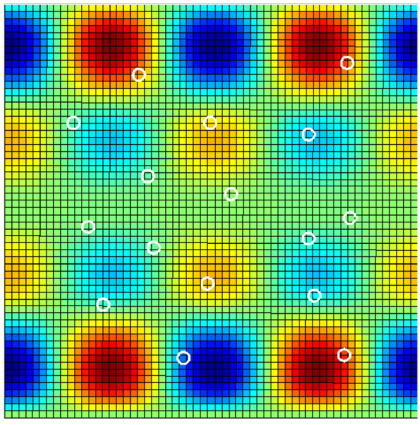
designed to adapt to the specific instance being executed and reduce the computation time as much as possible. To keep track of the number of real evaluations consumed, all candidate solutions share a counter, called *globalFuncEval*, throughout all iterations of the algorithm. This fact is also important when designing the parallel version. Figure 1 exemplifies the concept of a candidate solution in the search space.

OptiPharm requires four user-given parameters to be defined: the number of f.e. allowed for the whole optimization process (N), the number of solutions kept in the list (M), the number of iterations ($t_{\max}$), and the radius of the solutions created in the last iteration ($R_{t_{\max}}$). The parameter N is just an upper limit to bound the computational time. The higher this number, the greater the budget allowed for each candidate solution to reach the optimum. M has a high impact on the exploration of the search space, in the sense that the higher it is, the deeper the exploration will be (see Fig. 1). The value of $t_{\max}$ indicates how many times the reproduction-substitution-improvement sequence will take place during the optimization procedure. Finally, the radius is associated with each solution and conceptually connects the global and local exploration in OptiPharm. This radius is equal to the diameter of the search space in the initial solutions and decreases with each iteration of the algorithm up to a value specified by $R_{t_{\max}}$.

From an implementation point of view, the principal data structure of OptiPharm is a dynamic linked list in which elements are stored, deleted and updated. Depending on the particular phase of the algorithm, the list's elements can be considered as isolated points without any relation to each other or, on the contrary, as a set to be analyzed as a whole.



(a) Conceptualization example: Example of an objective function.



(b) Candidate solutions concept: The circles in white represent different candidate solutions in the search space.

Fig. 1 The procedures designed in OptPharm will direct candidate solutions toward promising areas of the search space, i.e., the more of them, the deeper the exploration. The higher the budget in the number of function evaluations per solution, the higher the chances of reaching the peak of the optimum

In the *Creating new poses* method, for each pose in the list, a set of potential new candidates is calculated and evaluated to find promising new areas, thus increasing the number of elements in the list. It is essential to mention that each individual generates new ones on its own, independently of the others. After that, if there is a surplus in the number of points in the population, the procedure *Elitist selection* orders the individuals according to their score value and eliminates those that are not the M best ones. Notice that this selection procedure requires knowledge of the entire population list. Finally, the method *Improvement* performs a local optimization on each solution, where the obtained local optimum replaces the called individual. Note that each point is optimized without taking into account the remaining ones. For a detailed description of the OptiPharm algorithm, see [14].

3 pOptiPharm: the parallel version

The problem addressed in this paper involves optimizing a particular query against a huge database. Two levels of parallelism can then be exploited. The first is related to the number of molecules in the database. From a conceptual point of view, this is an embarrassing parallelism problem, as it can be easily solved by launching OptiPharm processes to fully load a cluster. However, from a technical point of view, a considerable effort has to be made to automate this procedure and make it user-friendly for non-experts in the parallelism framework. Due to the scope of this journal, we do not provide further details on the implementation of the corresponding scripts, but note that they are available upon request for the interested reader/user.

Then, this section focus on the second layer of parallelism, which concerns OptiPharm's optimization of a single pair of compounds. In other words, the parallelization of Algorithm 1. Broadly speaking, our parallel approximation follows a shared memory model, in which the problem is accelerated by conceptually splitting the data into chunks that are later processed by different processing units.

From the implementation point of view, OptiPharm's main data structure is a linked list in which the poses are stored, sorted, and optimized. These operations are core parts of the algorithm, and thus it is of vital importance to make them as performant as possible. Those operations are applied in a locked-step, single-threaded process despite the self-reliance of the candidate solutions in the list. In fact, pOptiPharm's changes are mainly based on this property of the solutions. More precisely, pOptiPharm implements a custom thread-pool in order to help parallelize the work of the algorithm. This pool has two main operations: submit work and wait for all the work currently submitted to finish.

In the following, we will explain how the different steps of Algorithm 1 have been parallelized:

- *Initial poses creation* In the initialization phase, the query and the target are aligned and centered at the origin of the coordinates. In the sequential version, from this initial situation, a population of M poses is composed: (i) the first pose represents this initial stage, i.e., the query is not moved with respect to the target, which remains fixed, (ii) the following three poses are obtained by rotating the

variable molecule π radians at each axis (always from the initial state), the final $M-4$ poses are randomly computed to prevent a possible drift to local optima. For the parallel version, we have reduced the number of random poses to just one and increased the number of them generated from rotations of the variable molecule. In particular, we compute p poses, each rotated $\frac{2\pi}{p}$ radians in each axis, where $p = \max(\min(2, M), 13)$. Notice that 13 is an experimental value, i.e., reducing the angle more than 13 times produces no significant changes. During the creation of the initial poses, they have to be evaluated once before the rest of the algorithm can continue. In the original version, any initial candidate solution is evaluated right before being appended into the global poses list. In contrast, in the parallel version, the initial poses are first appended into the list, then evaluated in parallel using the thread-pool. The parallelization of this very step does not directly translate into a significant speed gain, but it helps to reduce the evaluation time of the initial poses and it allow us to increase the number of initial poses without increasing the execution time of the program.

- *New poses creation* During this step, new poses or *"children"* poses are created from the ones currently in the global poses list. For each one of them, one after another, *"children"* are generated into local linked (poses) lists that are later joined into the global list. Doing this same process in a multi-threaded environment requires us to block the global list every time a pose appends its *"children"* (see Fig. 2). Notice that, to make the process as fast and efficient as possible, the implemented algorithm has more synchronization and signaling than the ones indicated in the figure.
- *Elitist selection* Although this step is relatively simple in the sequential OptiPharm version, as it only consists on sorting the global poses list and then pruning the elements below the poses limit, in the parallel version this requires a paral-

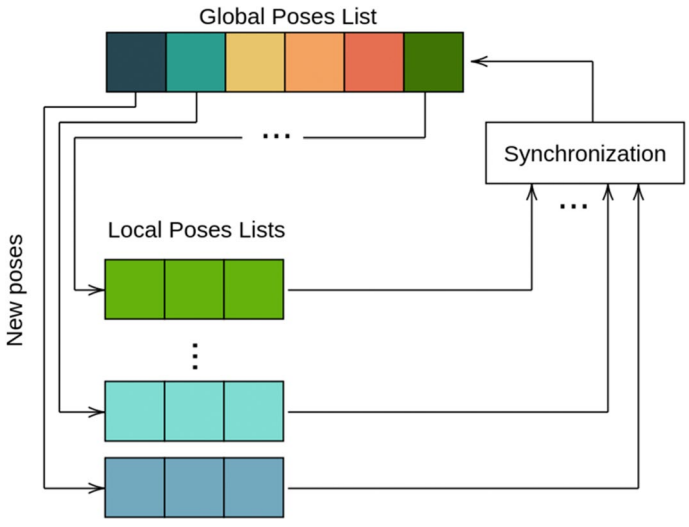


Fig. 2 Parallel new poses creation

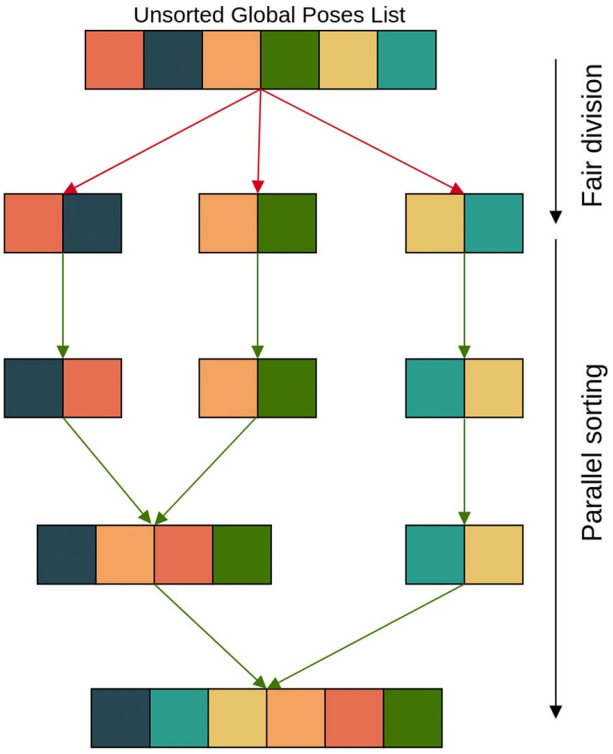


Fig. 3 Parallel merge sort

lel sorting method. For pOptiPharm, a slightly modified parallel merge sort has been implemented (see Fig. 3). After the sorting, the poses over the limit (M) are pruned in the same way as OptiPharm.

- *Optimization* At this stage of the algorithm, every pose in the list has to run through a custom version of an improving method. This is a very costly operation as this step runs the evaluation function a myriad of times for each single pose. To accelerate this process, in the parallel version, each pose in the global poses list runs the complete optimization process in a core provided by the thread-pool.

Finally, it is important to mention that due to the fact that all candidate solutions along all iterations of the algorithm share a counter for the current number of evaluations *globalFuncEval*, there has to be some synchronization to keep track of the evaluations in the parallel version correctly. For that, all the tasks submitted to the thread-pool share an atomic number which later is used to increment the global counter without causing any data race.

4 Experimental context

To implement the parallel version, C++ standard threading library `std::threads` (available from C++ 11 onwards) has been used. In the following, the framework for the computational experiments is explained. In particular, it is defined the objective function used to measure the similarity between two given molecules, the database, and hardware used in the experimentation, as well as the configurations of the different algorithms.

4.1 Shape similarity score

OptiPharm is a general-purpose algorithm that can solve any optimization problem involving the computation of the similarity of two input compounds. pOptiPharm maintains this important attribute. In this work, to illustrate the performance of the parallel version, we have solved a maximization problem that consists of finding the $\boldsymbol{s}$ solution that maximizes the Tc function defined in Eq. 1.

$$Tc(A,B) = \frac{V(A,B)}{V(A,A) + V(B,B) - V(A,B)} \tag{1}$$

It returns a value in the range [0, 1], where 0 indicates that there is no similarity between the input molecules A and B, and 1 implies that the similarity between the two molecules is maximal.

Each of the values $V(A, B)$, $V(A, A)$ and $V(B, B)$ is obtained by Eq. 2 where w_i and w_j are the weights represented by atom i and j of molecule A and B, respectively. On the other hand, v_{ij} is a product of Gaussian functions between atom i and j. This value represents the intersection of the volumes of these two atoms, so that by adding up all the intersections of the atoms, the total overlap value of molecule A with respect to molecule B (and vice versa) is calculated. For a better understanding of this scoring function widely used in the literature, we recommend reading the following papers [14, 16–18].

$$V(A,B) = \sum_{i \in A, j \in B} w_i w_j v_{ij} \tag{2}$$

Note that the computational cost of the scoring function depends on the number of atoms in each molecule, i.e., the more atoms there are, the more intersections have to be computed and thus the longer the computation time. To exemplify this in a real scenario, lets compare the following molecules: DB03754 and DB09159. They have 20 and 18 atoms, respectively. The time to evaluate the scoring function is 0.027 ms. On the other hand, molecules DB01337 and DB01339 have 101 and 98 atoms, and in this case, 0.699 ms were needed to obtain the similarity score. As the results will show, this is an important aspect that can affect the performance of the parallel version.

4.2 FDA database

The database used in this work is the DrugBank [19, 20]. This database contains approximately 10,000 compounds, of which 2037 are FDA-approved small molecule drugs. The FDA, U.S. Food and Drug Administration, is an American agency that is responsible for protecting and promoting public health by controlling, among other things, prescription and over-the-counter pharmaceutical drugs (medicines). It is a common practice [21], in the current scenario, to identify which pairs of compounds in the FDA database share a high degree of shape similarity.

In a comprehensive analysis to study the performance of the algorithms, a selection of query molecules has been performed. For that purpose, the database has been divided into three parts depending on the size of the molecules, i.e., small (1–49 atoms), medium (50–99), and large (100–801) as can be seen in Fig. 4. Then, three random molecules were chosen from each group and marked as queries. Specifically, in the group of small molecules, we randomly selected the compounds DB00920 (41 atoms), DB01124 (36) and DB03754 (20); as medium molecules we chose DB00479 (83), DB01130 (71) and DB09135 (56); finally, as large molecules we considered DB00970 (176), DB01282 (138) and DB01337 (101). Note that the number in brackets indicates the number of atoms in each molecule. Each molecule was pitted against the entire database to find the compound with the highest similarity.

4.3 OptiPharm configuration

Being parameterizable is one of the strengths of OptiPharm, as it can be tuned depending on whether the user prefers acceptable solutions with a reasonable computational time or, on the contrary, prefers to obtain better solutions at the cost of higher computational effort. With these two aims, several combinations of the parameters were tried in [14], to finally propose two main groups of input parameters:

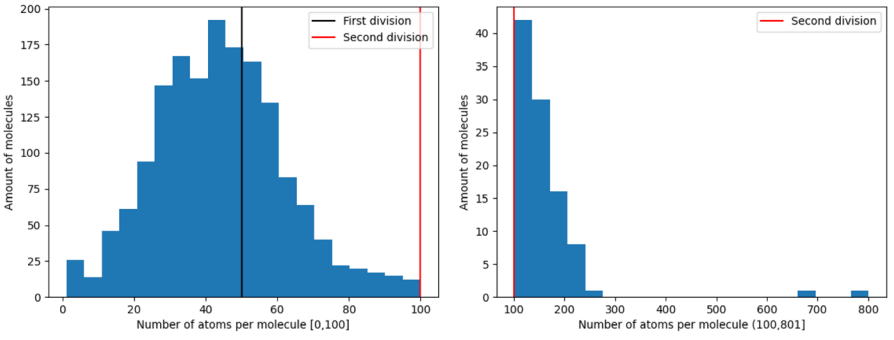


Fig. 4 Size distribution of molecules in the database. **a** shows the number of compounds with up to 100 atoms, and **b** shows the number of compounds with 100–801 atoms. The black and red lines indicate the division points of the 3 groups in the database, i.e., small, medium and large molecules (color figure online)

(i) *Fast configuration* (FC) The parameters are tuned so that OptiPharm runtime is small but reasonable solutions are obtained. The values considered for this configuration are: $N = 1000$ function evaluations, $M = 5$ starting poses, $t_{\max} = 5$ iterations and a minimum radius of $R_{t_{\max}} = 5$.
(ii) *Robust Configuration* (RC) This configuration parameters are tuned to make OptiPharm more reliable and robust, although it is more computationally expensive than the previous one: $N = 200{,}000$ function evaluations, $M = 5$ starting poses, $t_{\max} = 5$ iterations and a minimum radius of $R_{t_{\max}} = 1$.

However, with the use of parallel techniques, we can speed up the OptiPharm procedures without compromising the quality of the proposed solutions. This could be a huge step forward in the context of virtual screening, as most techniques in the literature do not sufficiently explore the search space due to the computational effort involved. In this study, we analyze OptiPharm's effectiveness by fine-tuning its parameters to obtain the best possible solution. To this end, building on the robust configuration proposed in the original paper [14], and referred to as RC(5) hereafter, three other robust configurations are proposed, each of which seeks to explore the search space more deeply:

(i) *Robust Configuration—15 poses* (RC (15)) This RC configuration parameters are: $N = 250{,}000$ function evaluations, $M = 15$ starting poses, $t_{\max} = 5$ iterations and a minimum radius of $R_{t_{\max}} = 1$.
(ii) *Robust Configuration—25 poses* (RC (25)) This RC configuration parameters are: $N = 370{,}000$ function evaluations, $M = 25$ starting poses, $t_{\max} = 5$ iterations and a minimum radius of $R_{t_{\max}} = 1$.
(iii) *Robust Configuration—50 poses* (RC (50)) This RC configuration parameters are: $N = 500{,}000$ function evaluations, $M = 50$ starting poses, $t_{\max} = 5$ iterations and a minimum radius of $R_{t_{\max}} = 1$.

Basically, we have increased the number of candidate points M and the total number of function evaluations N, with the aim of finding higher quality solutions.

4.4 Hardware setup

All the experiments carried out in this work have been executed in a Bull Sequana X440-A5, which consists of 2 AMD EPYC Rome 7642 (48 cores), 512 GB RAM and 240 GB SSD.

5 Results

This section shows the quality results of the OptiPharm and its parallel version pOptiPharm, as well as the speed-up of the experiments. It is important to mention that, due to the stochastic nature of both algorithms, the results gathered for the following experiments are computed from a 15-tries average (Fig. 5).

5.1 Qualitative analysis

For a better understanding of the numbers, we will initially focus on a single case, namely the query molecule DB09135. This query has been confronted with all the target compounds in the FDA database. More specifically, for each pair of molecules, OptiPharm has been run looking for the rotation and translation of the target that achieves the maximum overlap with the query. All the configurations listed in Sect. 4.3 have been considered. With configuration FC, we found that the molecule with the highest similarity was DB08947 with a *Tc* value of 0.68027. In contrast, with the robust configurations, OptiPharm has achieved increasingly higher similarity values, from 0.71610 with RC (5) up to 0.71645 with RC (50). Of course, this increase comes at the cost of a longer calculation time, whereas with FC configuration, OptiPharm spends 0.04646 s, it needs 21.50573 s with RC (50).

Similar analyses have been performed for the set of molecules listed in Sect. 4.2. More specifically, for each query, the best compound with the highest *Tc* is obtained. The corresponding results have been summarized in two complementary tables. On the one hand, Table 1 shows the value of *Tc* obtained by OptiPharm when run with all the configurations indicated in Sect. 4.3. Note that the results are displayed in three groups based on the size of the molecules. For each group, we have calculated the mean of the *Tc* values. Finally, the mean of the averages is also calculated and shown in the last row (Total Av). On the other hand, Table 2 shows the set of molecules (more than one can be obtained) from the database selected as the best compounds, and whose similarity score *Tc* is that of Table 1.

A detailed analysis of Table 1 shows that there is a first leap in the quality of the solutions between FC and RC configurations. However, focusing on the last three columns, especially on the average values (Av), it can be observed that although the differences are much smaller in these columns, there exists an increasing

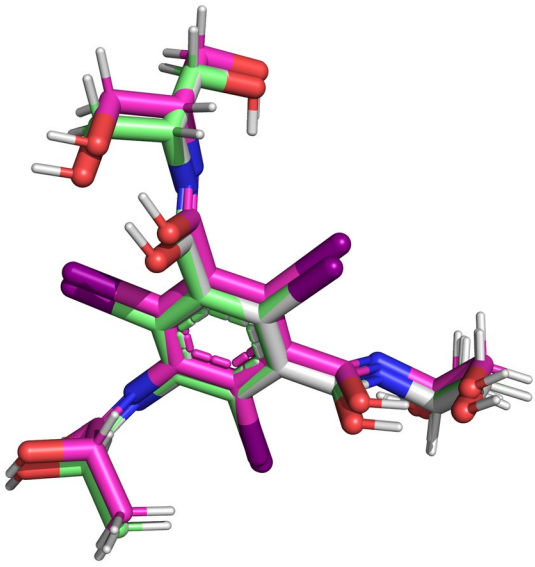


Fig. 5 Comparison of poses for target DB08947 and query DB09135. The figure shows the target in three different poses. In pink is represented the solution obtained by OptiPharm when running with the FC configuration, in green, the one obtained when considering RC (5) and in white the one obtained by RC (50). As can be seen, there are differences in position among the three solutions, which highlights the importance of achieving better optimization (color figure online)

Table 1 *Tc* value obtained for OptiPharm with the different parameter configurations

	FC	RC (5)	RC (15)	RC (25)	RC (50)
DB00920	0.87934	0.90577	0.90577	0.90578	0.90578
DB01124	0.91306	0.92158	0.92160	0.92160	0.92160
DB03754	0.89745	0.90058	0.90075	0.90075	0.90075
Av_{small}	0.89662	0.90931	0.90937	0.90938	0.90938
DB00479	0.64120	0.67618	0.67618	0.67618	0.67618
DB01130	0.69204	0.71956	0.72036	0.72036	0.72059
DB09135	0.68027	0.71610	0.71610	0.71637	0.71645
Av_{medium}	0.67117	0.70395	0.70421	0.70430	0.70441
DB00970	0.48769	0.49854	0.49876	0.49883	0.49895
DB01282	0.52667	0.55137	0.55137	0.55137	0.55137
DB01337	0.73025	0.73979	0.91139	0.91139	0.91140
Av_{large}	0.58154	0.59657	0.65384	0.65386	0.65391
Total Av	0.71644	0.73661	0.75581	0.75585	0.75590

improvement. Other interesting results appears in Table 2. As can be seen, OptiPharm with the robust configurations is able to provide, in some cases, more than one molecule with the maximum value of *Tc*. As an example, for DB03754, FC obtains as the best compound the molecule DB09159 with a $Tc = 0.89745$, but all the RC configurations obtain besides DB09159, the molecule DB09154. These results can have a major impact on the virtual selection process, as it offers more than one equally valid alternative. As an illustration, we show in Fig. 6 this concrete example using VIDA [22], i.e., the overlap between the query DB03754 and the two molecules DB09159 and DB09154. It is important to mention that for the molecule *DB*01282 two isolated cases exist where the parallel version only offers a single solution, i.e., for RC (15) and RC (50). In those cases, the second molecule with the

Table 2 Molecules obtained as the best compound by OptiPharm with the different parameter settings

	FC	RC (5)	RC (15)	RC (25)	RC (50)
DB00920	DB00719	DB00719	DB00719	DB00719	DB00719
DB01124	DB00672	DB00672	DB00672	DB00672	DB00672
DB03754	DB09159	DB09159 DB09154	DB09159 DB09154	DB09159 DB09154	DB09159 DB09154
DB00479	DB00288	DB00684	DB00684	DB00684	DB00684
DB01130	DB00596	DB01340	DB01340	DB01340	DB01340
DB09135	DB08947	DB08947	DB08947	DB08947	DB08947
DB00970	DB08890 DB06287	DB08890 DB06287	DB08890 DB06287	DB06287 DB08890	DB06287 DB08890
DB01282	DB00864	DB01211 DB06439	DB01211	DB06439 DB01211	DB06439
DB01337	DB00728	DB00728	DB01339	DB01339	DB01339

best *Tc* value was DB06439 (the missing molecule in the table). It can happen due to the stochastic nature of the algorithm.

These valuable results can be explained by the fact that, in general, OptiPharm is designed to maintain population diversity and investigate many promising solutions in parallel, avoiding genetic drift toward a single (local or global) optimal position. However, depending on the set of parameters selected, the accuracy in approximating the optima may be higher or lower. Of course, the larger the values of M and N, the greater the computational effort allowed and, consequently, the better the accuracy and the scoring results.

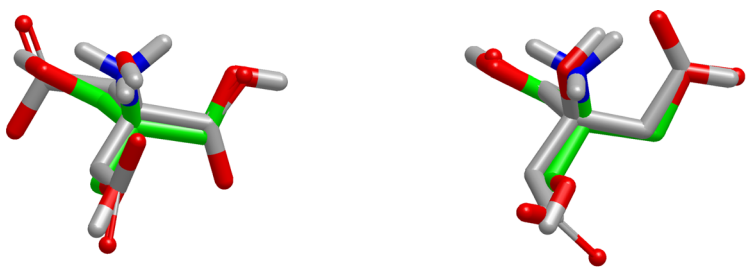
All this means that better solutions can be accomplished by using robust configurations when there are no time constraints. However, if lower execution times are required, using high-performance computing approaches, we could save computational time without giving up an improvement in the quality of the solution.

5.2 Performance analysis

Next, we analyze the performance of the parallel version when run with all the configurations listed in Sect. 4.3. From an efficiency point of view, pOptiPharm with all the configurations offer the same quality results as those indicated in Tables 1 and 2. We then focus on their efficiency.

The virtual screening problem we are dealing with is terribly irregular, in the sense that solving a particular instance involves running a huge number of optimization problems (as many as molecules in the database), all of them with a different computational burden. Therefore, to understand the results, we need to analyze partial results.

Table 3 refers to the speed-up values obtained by pOptiPharm with RC (15). For this case, we have faced the query against the whole database and computed the speed-up for each query-target pair. However, for the sake of brevity and clarity, we have extracted some partial results. In particular, the speed-up obtained



(a) Target molecule: DB09159. $Tc = 0.90058$ (b) Target molecule: DB09154. $Tc = 0.90058$

Fig. 6 For the query DB03754, two different solutions has been found by OptiPharm with RC (5). The query molecule, painted in green, remains fixed in all subfigures (color figure online)

when the target is the molecule with the highest similarity value *Tc*, i.e., the one shown in Table 2. This will be identified by the symbol "∗". Furthermore, we have selected the target with the best speed-up value for 32 threads (denoted with "↑") and also the one with the worst value (denoted with "↓"). For these three targets, we show their number of atoms in the column *A*. Finally, to have a global view of the whole problem, we have calculated the speed-up for the entire database. This result is indicated in the table with the symbol "

". In this case, the column *A* refers to the number of atoms, on average, in the whole database. The speed-up has been calculated for th $= 2, 4, 8, 16, 32$ threads and shown in Spd(th) columns. For completeness, the computation time *Ts*(1) spent by OptiPharm (the sequential algorithm) when it is run with RC (15) configuration, is also provided.

A detailed analysis of Table 3 shows that pOptiPharm scales almost linearly up to 16 threads for the entire database (see row in bold associated to

), independently of the query considered. Something similar happens for the pair with the best similarity (*) and the best speed-up (↑). In fact, up to 8 threads, a super speed-up is obtained in some cases. On the contrary, comparing any of the previously mentioned pairs against the one with the worst speed-up (↓), we can clearly see the existing in-balanced, which affects the algorithm workload and makes speed-up less than optimal. It happens because the speed-up of this problem is molecule-size dependent. As mentioned when describing the objective function in Sect. 4.1, more atoms make the objective function more expensive. Consequently, if there are not enough atoms, the computational cost of creating and managing threads is more time-consuming than the execution of the scoring function itself. In this table we can also see that adding more than 16 threads poorly contributes to improving the speed-up, even causing speed-up loss when there are more threads than work available.

Similar studies were carried out for the RC (25) and RC (50) configurations, but to save time, only th $= 2, 8, 32$ were considered. The conclusions drawn were similar to those highlighted above. So, for the sake of brevity, we do not include those tables, but a summary that includes the speedup results for the whole database, i.e., the one corresponding to the

line. That is Table 4.

As seen in the table, pOptiPharm shows a linear speed up for a small number of threads. On the contrary, for great values, it does not reach the ideal one. Nevertheless, we can observe scalability, i.e., the speed-up increases as the configurations are more demanding, rising from 14.2 for RC (15) to 24.3 when RC (50) is considered. This trend is similar for all the molecules studied.

pOptiPharm behaves as expected, since its parallelism comes from the *M* number of solutions in the population list. We have also observed that the speedup does not increase proportionally from the RC (25) to the RC (50) configuration. This is because the number *N* of function evaluations in the configurations does not increase proportionally to the number *M* of poses in the list, so the budget (number of function evaluations per solution) does not grow proportionally either. On the contrary, we have increased it based on our experience and following the guidelines provided in [14, 23] to obtain higher quality solutions. If we had done so, we could have improved the trend experienced by pOptiPharm in terms of speedup for 32 threads or more, but we would

Table 3 Speed-up analysis for RC (15)

Query	Target	A	$T_s(1)$	Spd (2)	Spd (4)	Spd (8)	Spd (16)	Spd (32)
DB00920	* DB00719	44	8.4	2.1	4.3	8.5	16.4	15.4
DB00920	↑ DB01111	212	38.7	2.2	4.5	8.9	17.5	16.7
DB00920	↓ DB09315	1	0.2	0.8	1.7	3.3	4.5	3.7
DB00920	🐾	50	15262.2	2.0	4.0	7.9	15.3	14.2
DB01124	* DB00672	30	5.0	2.0	4.1	8.1	14.9	13.3
DB01124	↑ DB06402	228	36.0	2.2	4.4	8.7	17.2	16.3
DB01124	↓ DB09315	1	0.2	0.8	1.6	3.2	4.0	3.2
DB01124	🐾	50	13609.4	2.0	4.0	7.9	15.1	13.5
DB03754	* DB09159	18	1.8	1.9	3.8	7.5	13.6	11.4
DB03754	↑ DB00159	52	5.4	2.3	4.7	9.3	17.9	16.9
DB03754	↓ DB09315	1	0.1	0.6	1.3	2.6	3.2	2.8
DB03754	🐾	50	7903.6	2.0	4.0	7.9	15.2	13.9
DB00479	* DB00684	71	25.6	2.2	4.4	8.6	16.8	16.1
DB00479	↑ DB00803	179	65.4	2.2	4.5	8.9	17.5	17.2
DB00479	↓ DB09315	1	0.3	1.1	2.2	4.3	6.5	6.1
DB00479	🐾	50	29875.3	2.0	4.0	7.9	15.4	14.5
DB01130	* DB01340	61	19.3	2.2	4.3	8.5	16.5	15.8
DB01130	↑ DB01193	53	18.2	2.4	4.7	9.3	18.1	16.5
DB01130	↓ DB09315	1	0.3	1.04	2.08	4.07	4.2	4.6
DB01130	🐾	50	25745.2	2.0	4.0	7.9	15.3	14.1
DB09135	* DB08947	53	13.3	2.2	4.3	8.6	16.7	15.2
DB09135	↑ DB06813	56	15.2	2.4	4.7	9.4	18.3	16.3
DB09135	↓ DB09315	1	0.2	0.9	1.9	3.7	5.1	3.9
DB09135	🐾	50	20455.6	2.0	4.0	7.9	15.3	14.0
DB00970	* DB08890	184	137.4	2.2	4.3	8.7	16.9	16.5
DB00970	↑ DB01337	101	78.6	2.2	4.5	9.0	17.8	17.3
DB00970	↓ DB09315	1	0.6	1.4	2.7	5.4	9.4	8.1
DB00970	🐾	50	26531.9	2.0	4.0	7.9	15.5	14.9
DB01282	* DB01211	122	74.1	2.2	4.5	8.8	17.3	17.0
DB01282	↑ DB01336	96	64.3	2.4	4.9	9.7	19.0	18.6
DB01282	↓ DB09315	1	0.5	1.3	2.5	5.1	8.7	6.4
DB01282	🐾	50	26452.9	2.0	4.0	7.9	15.5	14.7
DB01337	* DB01339	98	44.0	2.2	4.4	8.7	17.0	16.4
DB01337	↑ DB00644	160	71.1	2.2	4.5	9.0	17.7	17.0
DB01337	↓ DB09315	1	0.4	1.1	2.3	4.6	7.5	6.2
DB01337	🐾	50	26540.9	2.0	4.0	7.9	15.4	14.6

have overestimated the budget per point looking for an improvement in speedup and not looking for a balance between time and quality of the solution found. Another fact, that may explain the drop in the speed-up for 32 threads is that the parallel version has several synchronization points to maintain consistency with the sequential version. The effect of these synchronization points is greater as the number of threads increases.

Table 4 Speed-up summary for RC (15), RC (25) and RC (50)

Query	RC	*Ts*(1)	Spd (2)	Spd (8)	Spd (32)
DB00920	RC (15)	15262.2	2.0	7.9	14.2
DB00920	RC (25)	19959.4	2.0	7.9	22.8
DB00920	RC (50)	27213.8	2.0	7.9	24.3
DB01124	RC (15)	13609.4	2.0	7.9	13.5
DB01124	RC (25)	17941.1	2.1	7.9	17.6
DB01124	RC (50)	24314.9	2.0	7.8	20.7
DB03754	RC (15)	7903.6	2.0	7.9	13.9
DB03754	RC (25)	10336.1	2.1	7.8	21.3
DB03754	RC (50)	14052.7	2.0	7.8	24.1
DB00479	RC (15)	29875.3	2.0	7.9	14.5
DB00479	RC (25)	39065.3	2.0	7.8	23.7
DB00479	RC (50)	53307.4	2.0	7.9	24.4
DB01130	RC (15)	25745.2	2.0	7.9	14.1
DB01130	RC (25)	33934.1	2.1	7.9	19.3
DB01130	RC (50)	46066.3	2.0	7.6	22.0
DB09135	RC (15)	20455.6	2.0	7.9	14.0
DB09135	RC (25)	26979.0	2.1	7.8	18.7
DB09135	RC (50)	36599.1	2.0	7.5	21.6
DB00970	RC (15)	26531.9	2.0	7.9	14.9
DB00970	RC (25)	34709.5	2.0	7.7	24.7
DB00970	RC (50)	47382.9	2.0	7.9	24.4
DB01282	RC (15)	26452.9	2.0	7.9	14.7
DB01282	RC (25)	34837.5	2.1	7.8	21.0
DB01282	RC (50)	47334.0	2.0	7.8	23.1
DB01337	RC (15)	26540.9	2.0	7.9	14.6
DB01337	RC (25)	34688.9	2.0	7.8	23.9
DB01337	RC (50)	47318.3	2.0	7.8	24.4

Once the efficiency results are analyzed, we can conclude that pOptiPharm is a good alternative to speed-up the virtual screening process, while improving the quality of the solutions provided.

6 Conclusions

A new algorithm called OptiPharm was recently proposed to accelerate drug discovery processes. Experiments showed that OptiPharm was reliable in terms of prediction accuracy and runtime, and very competitive compared to approaches previously described in the literature.

One of the main strengths of OptiPharm is that it is easily parameterizable, so that very different configuration schemes can be tested. In this work, we have explored this fact, finding configurations that improve the provided solution, not only in terms

of accuracy but, more importantly, in terms of the number of solutions with maximum similarity. To address this drawback, we have proposed a high-performance computing approach, named pOptiPharm. Parallelism in pOptiPharm has been applied at two levels: (i) by automatically distributing the database molecules on the different nodes of the cluster and (ii) by exploiting the intrinsic parallelism of OptiPharm, basically by dividing the number of solutions among the available processing units to perform the replication, selection and optimization operations individually.

Computational experiments have shown that the size of the molecules strongly influences the accelerations obtained. Thus, pOptiPharm is able to obtain ideal or super-ideal accelerations for large molecules, while it obtains very low values for small compounds. These results are as expected since the computational load must be sufficient to compensate for the parallelism overhead. However, since the databases are composed of molecules of different sizes, the final speed-up will depend on the number of molecules of each size included in each database.

Focusing on those cases with a sufficiently high computational load, i.e., those instances with molecules with a large number of atoms, we can see that good speed-up results have been achieved, mainly up to $\mathrm{th} = 16$ threads, where an almost linear or even superlinear speed-up is obtained. For larger values of th we do not obtain ideal values, but we still obtain good speed-ups. Finally, it is important to mention that pOptiPharm scales and obtains better efficiency results as the computational load increases.

Considering all these results, pOptiPharm is a good alternative to OptiPharm regarding solution quality and computational cost savings. The latter is significant for two main reasons: (i) other descriptors are used as objective functions that require more time, and (ii) the databases to be processed are enormously large. This contribution is not only relevant in the current context, where molecules are considered rigid, but may become even more critical in VS problems where molecules are flexible, as the databases are even more extensive (see [24–27]).

In future, other programming paradigms based on both shared and distributed memory architectures will be implemented and analyzed. In particular, a parallel version of OptiPharm will be implemented to be executed on GPUs. Additionally, pOptiPharm will be incorporated into the BRUSELAS platform [28], where the sequential version is also available so that any user can freely execute the code. In the meantime, the interested reader can obtain the code on request. Finally, pOptiPharm will be applied to solve the problem where the molecules are considered flexible.

Author contributions All authors contributed equally to this work. JSG, SPM, JLR, JJM and PMO conceived the experiments, conducted the experiments, analyzed the results and reviewed the manuscript.

Funding This work has been financed by the National R+D+i Plan Project PID2021-123278OB-I00 of the Spanish Ministry of Science and Innovation and EIE funds; the Andalusian Regional Government through the grant: Proyectos de Excelencia (P18-RT-1193); and finally, the University of Almería through the grant: "Ayudas a proyectos de investigación I+D+I en el marco del Programa Operativo FEDER 2014–20" (UAL18-TIC-A020-B). Savíns Puertas Martín is a fellow of the "Margarita Salas" grant (RR_A_2021_21), financed by the European Union (NextGenerationEU). J.J. Moreno is supported by an FPU Fellowship (FPU16/05946) from the Spanish Ministry of Education.

Data availability The databases and software belong to their authors and access to them depends on any applicable restrictions.

Declarations

Conflict of interest The authors declare no competing interests.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Xu X, Chen P, Wang J, Feng J, Zhou H, Li X, Zhong W, Hao P (2020) Evolution of the novel coronavirus from the ongoing Wuhan outbreak and modeling of its spike protein for risk of human transmission. Sci China Life Sci 63:457–460. https://doi.org/10.1007/s11427-020-1637-5
2. Sheinson DM, Wong WB, Solon CE, Cheng MM, Shah A, Elsea D, Meng Y (2021) Estimated impact of public and private sector Covid-19 diagnostics and treatments on us healthcare resource utilization. Adv Ther 38:1212–1226. https://doi.org/10.1007/s12325-020-01597-3
3. Hughes J, Rees S, Kalindjian S, Philpott K (2011) Principles of early drug discovery. Br. J. Pharmacol. 162(6):1239–1249. https://doi.org/10.1111/j.1476-5381.2010.01127.x
4. Morgan S, Grootendorst P, Lexchin J, Cunningham C, Greyson D (2011) The cost of drug development: a systematic review. Health Policy 100:4–17. https://doi.org/10.1016/j.healthpol.2010.12.002
5. Wildey MJ, Haunso A, Tudor M, Webb M, Connick JH (2017) High-throughput screening. Annu Rep Med Chem. https://doi.org/10.1016/bs.armc.2017.08.004
6. Shen L, Niu J, Wang C, Huang B, Wang W, Zhu N, Deng Y, Wang H, Ye F, Cen S, Tan W (2019) High-throughput screening and identification of potent broad-spectrum inhibitors of coronaviruses. J Virol. https://doi.org/10.1128/JVI.00023-19
7. Zeng W, Guo L, Xu S, Chen J, Zhou J (2020) High-throughput screening technology in industrial biotechnology. Trends Biotechnol 38:888–906. https://doi.org/10.1016/j.tibtech.2020.01.001
8. Fischer A, Sellner M, Neranjan S, Smieško M, Lill MA (2020) Potential inhibitors for novel coronavirus protease identified by virtual screening of 606 million compounds. Int J Mol Sci 21:3626. https://doi.org/10.3390/ijms21103626
9. Maia EHB, Assis LC, de Oliveira TA, da Silva AM, Taranto AG (2020) Structure-based virtual screening: from classical to artificial intelligence. Front Chem. https://doi.org/10.3389/fchem.2020.00343
10. Ripphausen P, Nisius B, Bajorath J (2011) State-of-the-art in ligand-based virtual screening. Drug Discov Today 16:372–376. https://doi.org/10.1016/j.drudis.2011.02.011
11. Glantz-Gashai Y, Meirson T, Reuveni E, Samson AO (2017) Virtual screening for potential inhibitors of mcl-1 conformations sampled by normal modes, molecular dynamics, and nuclear magnetic resonance. Drug Des Dev Ther 11:1803–1813. https://doi.org/10.2147/DDDT.S133127
12. Lyne PD (2002) Structure-based virtual screening: an overview. Drug Discov Today 7:1047–1055. https://doi.org/10.1016/S1359-6446(02)02483-2
13. dos Santos RN, Ferreira LG, Andricopulo AD (2018) Practices in molecular docking and structure-based virtual. Screening. https://doi.org/10.1007/978-1-4939-7756-7_3
14. Puertas-Martín S, Redondo JL, Ortigosa PM, Pérez-Sánchez H (2019) OptiPharm: an evolutionary algorithm to compare shape similarity. Sci Rep 9(1):1398. https://doi.org/10.1038/s41598-018-37908-6
15. Puertas-Martín S, Redondo LJ, Pérez-Sánchez H, Ortigosa MP (2020) Optimizing electrostatic similarity for virtual screening: a new methodology. Informatica. https://doi.org/10.15388/20-INFOR424
16. Yan X, Li J, Liu Z, Zheng M, Ge H, Xu J (2013) Enhancing molecular shape comparison by weighted Gaussian functions. J Chem Inf Model 53(8):1967–1978. https://doi.org/10.1021/ci300601q

17. Grant JA, Pickup BT (1995) A Gaussian description of molecular shape. J Phys Chem 99(11):3503–3510. https://doi.org/10.1021/j100011a016
18. Grant JA, Gallardo MA, Pickup BT (1996) A fast method of molecular shape comparison: a simple application of a Gaussian description of molecular shape. J Comput Chem 17(14):1653–1666
19. Wishart DS (2006) DrugBank: a comprehensive resource for in silico drug discovery and exploration. Nucl Acids Res 34(90001):668–672. https://doi.org/10.1093/nar/gkj067. arXiv:arXiv:1011.1669v3
20. Wishart DS, Feunang YD, Guo AC, Lo EJ, Marcu A, Grant JR, Sajed T, Johnson D, Li C, Sayeeda Z, Assempour N, Iynkkaran I, Liu Y, Maciejewski A, Gale N, Wilson A, Chin L, Cummings R, Le D, Pon A, Knox C, Wilson M (2018) DrugBank 5.0: a major update to the DrugBank database for 2018. Nucl Acids Res 46(D1):1074–1082. https://doi.org/10.1093/nar/gkx1037
21. Den-Haan H, José Hernández Morante J, Pérez-Sánchez H (2016) Computational evidence of a compound with nicotinic $\alpha 4\beta 2$-ach receptor partial agonist properties as possible coadjuvant for the treatment of obesity. bioRxiv. https://doi.org/10.1101/088138
22. OpenEye Scientific Software, Software, O.S., OpenEye Scientific Software: VIDA 4.4.0.4 (2019). www.eyesopen.com
23. Ortigosa PM, García I, Jelásity M (2001) Reliability and performance of UEGO, a clustering-based global optimizer. J Glob Optim 19(3):265–289. https://doi.org/10.1023/A:1011224107143
24. Kalászi A, Szisz D, Imre G, Polgár T (2014) Screen3D: a novel fully flexible high-throughput shape-similarity search method. J Chem Inf Model 54(4):1036–1049. https://doi.org/10.1021/ci400620f
25. Hu J, Liu Z, Yu DJ, Zhang Y (2018) LS-align: an atom-level, flexible ligand structural alignment algorithm for high-throughput virtual screening. Bioinformatics 34(13):2209–2218. https://doi.org/10.1093/bioinformatics/bty081
26. Ferraz WR, Gomes RA, S Novaes AL, Goulart Trossini GH (2020) Ligand and structure-based virtual screening applied to the sars-cov-2 main protease: an in silico repurposing study. Fut Med Chem 12(20):1815–1828. https://doi.org/10.4155/fmc-2020-0165
27. Fischer A, Sellner M, Neranjan S, Smieško M, Lill MA (2020) Potential inhibitors for novel coronavirus protease identified by virtual screening of 606 million compounds. Int J Mol Sci. https://doi.org/10.3390/ijms21103626
28. Banegas-Luna AJ, Cerón-Carrasco JP, Puertas-Martín S, Pérez-Sánchez H (2019) Bruselas: Hpc generic and customizable software architecture for 3d ligand-based virtual screening of large molecular databases. J Chem Inf Model 59:2805–2817. https://doi.org/10.1021/acs.jcim.9b00279

Authors and Affiliations

Jerónimo S. García[1] · Savíns Puertas-Martín[1,2] · Juana L. Redondo[1] · Juan José Moreno[1] · Pilar M. Ortigosa[1]

Jerónimo S. García
jeronimosg@ual.es

Savíns Puertas-Martín
savinspm@ual.es

Juan José Moreno
jjmoreno@ual.es

Pilar M. Ortigosa
ortigosa@ual.es

[1] Supercomputing - Algorithms Research Group (SAL), Agrifood Campus of International Excellence, University of Almería, Carretera Sacramento s/n, La Cañada de San Urbano, 04120 Almería, Spain

[2] Information School, University of Sheffield, 221, Portobello Street, Sheffield S1 4DP, United Kingdom