Bioinformatics, 2020, 1–6
doi: 10.1093/bioinformatics/btz958
Advance Access Publication Date: 21 January 2020
Original Paper

Structural bioinformatics

METADOCK 2: a high-throughput parallel metaheuristic scheme for molecular docking

Baldomero Imbernón*, Antonio Serrano, Andrés Bueno-Crespo , José L. Abellán, Horacio Pérez-Sánchez* and José M. Cecilia*

Structural Bioinformatics and High Performance Computing Research Group (BIO-HPC), Computer Engineering Department, Universidad Católica de Murcia (UCAM), 30107 Murcia, Spain

*To whom correspondence should be addressed.

Associate Editor: Alfonso Valencia

Received on January 15, 2019; revised on November 2, 2019; editorial decision on December 19, 2019; accepted on January 16, 2020

Abstract

Motivation: Molecular docking methods are extensively used to predict the interaction between protein–ligand systems in terms of structure and binding affinity, through the optimization of a physics-based scoring function. However, the computational requirements of these simulations grow exponentially with: (i) the global optimization procedure, (ii) the number and degrees of freedom of molecular conformations generated and (iii) the mathematical complexity of the scoring function.

Results: In this work, we introduce a novel molecular docking method named *METADOCK* 2, which incorporates several novel features, such as (i) a ligand-dependent blind docking approach that exhaustively scans the whole protein surface to detect novel allosteric sites, (ii) an optimization method to enable the use of a wide branch of metaheuristics and (iii) a heterogeneous implementation based on multicore CPUs and multiple graphics processing units. Two representative scoring functions implemented in *METADOCK* 2 are extensively evaluated in terms of computational performance and accuracy using several benchmarks (such as the well-known DUD) against AutoDock 4.2 and AutoDock Vina. Results place *METADOCK* 2 as an efficient and accurate docking methodology able to deal with complex systems where computational demands are staggering and which outperforms both AutoDock Vina and AutoDock 4.

Availability and implementation: https://Baldoimbernon@bitbucket.org/Baldoimbernon/metadock_2.git.

Contact: bimbernon@ucam.edu or hperez@ucam.edu or jmcecilia@ucam.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

The advent of computational drug discovery (CDD) methods has democratized and accelerated the discovery of new drugs by developing tools, such as molecular docking to simulate the interaction between small molecules with potential inhibitory capacity and protein targets (Sliwoski *et al.*, 2014). This process is mandatory due to the effects of globalization, where emerging diseases may provoke pandemics worldwide and health institutions must provide treatments quickly. Some recent examples of this include Zika (Yuan *et al.*, 2017) and Ebola virus (Sakurai *et al.*, 2015).

The success of CDD methods is limited by the computational requirements they demand. The fastest docking methods are not able to process the largest biological databases in a reasonable timeframe, which actually limits their practical application in medical research. The use of high performance computing to speed-up CDD methods is therefore necessary to fulfill pharmaceutical industry expectations. The molecular docking methods available in the literature, such as Autodock (Morris *et al.*, 1998), Glide (Friesner *et al.*, 2004) or DOCK (Ewing *et al.*, 2001) are mainly designed to be executed in large homogeneous clusters of CPUs using Message Passing Interface with multithreading programing techniques at the node level to use the multiple cores of current CPUs. However, we are currently witnessing a steady transition to heterogeneous computing systems (http://www.top500.org/), with heterogeneity representing systems where nodes combine traditional multicore architectures (CPUs) with accelerators, such as graphics processing units (GPUs). Indeed, some molecular docking methods are being recently developed to fully leverage heterogeneous systems, such as BUDE (McIntosh-Smith *et al.*, 2015), BINDSURF (Sánchez-Linares *et al.*, 2012) or METADOCK (Imbernón *et al.*, 2017). The performance of these GPU-based methods places them as very competitive molecular docking tools. Moreover, certain efforts have been made in the World Community Grid on many thousands of desktop machines

with AutoDock Vina but which have no GPU implementation available to speed it up. This has actually been a promising area of research for many years now which is also indicative of the enormous demand that there is for ligand screening and the CPU time as a bottleneck (Guerrero *et al.*, 2014).

However, the implementation of docking methods in GPUs is not straightforward. GPUs are massively parallel by definition (http://developer.nvidia.com) and they require the redefining and even rethinking of the underlying algorithm to fully leverage their horsepower (Dakkak *et al.*, 2018). Most of the docking methods were based on optimization procedures, such as Monte Carlo and simulated annealing, which were implemented in a sequential fashion. In our previous work, we designed *METADOCK* (*METADOCK* 1.0) (Imbernón *et al.*, 2017), a molecular docking methodology based on a parameterized metaheuristics schema. The underlying scoring function of *METADOCK* was only based on the Lennard-Jones term but, instead of having a particular optimization algorithm, *METADOCK* provided a framework to choose the optimization procedure at runtime by choosing a set of input parameters. All the optimization procedures chosen were within the umbrella of population-based metaheuristics (Blum and Roli, 2003), such as genetic algorithms, which generate initial populations of individuals (or candidate solutions) that are iteratively improved in parallel. Population-based metaheuristics are inherently parallel by definition and therefore are well-suited for massive parallelization in the current landscape of computation (Cecilia *et al.*, 2018).

In this work, we introduce *METADOCK* 2, an improved version of our previous work *METADOCK* (Imbernón *et al.*, 2017). The earlier version was designed from scratch and powered by the current landscape of computation where the convergence between high performance computing and AI is pushing forward the frontiers of computation. The main new features included in *METADOCK* 2 and thus the contributions of this paper can be summarized as follows:

1. *METADOCK* 2 improves the scoring function of its predecessor by implementing two different versions that are deeply analyzed. The former was based on Autodock 4.2 and chosen as such to enable direct comparison with its prediction accuracy and also because it is one of the most widely used pieces of docking software, while the latter is based on the scoring function included in BINDSURF (Sánchez-Linares *et al.*, 2012).
2. The internal flexibility of the ligand is implemented in terms of rotatable bonds.
3. *METADOCK* 2 includes new features in the metaheuristic schema so as to be able to generate a larger number of metaheuristics and hybridizations than the previous version.
4. The predictive accuracy of this method is evaluated by redocking all protein–ligand crystal complexes in the DUD dataset (Mysinger *et al.*, 2012) and also performing blind docking (BD) on difficult systems (either too many rotatable bonds or very large proteins), such as protein–peptide (flexible) complexes from the LEADS-PEP dataset (Hauser and Windshügel, 2016) and protein systems with 100 000 atoms.
5. *METADOCK* 2 is compared in terms of performance and predictive accuracy with two state-of-the-art docking methods, namely, AutoDock Vina and AutoDock 4.2. Our results conclude that *METADOCK* 2 is a convincing alternative for dealing with complex systems where computation is a limiting factor.

The rest of the paper is structured as follows. Section 2 shows the insights of METADOCK 2 divided into two main categories: the search method based on a metaheuristic schema and the scoring function developed in the application. Next, the main results are discussed in Section 3 before we summarize the conclusions and provide some directions for future work.

2 Materials and methods

METADOCK 2 aims to predict the *binding-conformation* and affinity estimation of small ligand molecules (*ligand*) to potential protein targets. The computational representation of these molecular systems is based on the description of their atoms and it includes the following information: Cartesian coordinates for translation (x, y, z), ligand orientation through quaternions and a characterization of ligand torsions. Once this information is loaded into memory, the simulation is performed through an iterative process that aims to minimize a particular *scoring function* to obtain a set of candidate binding modes (*poses*)—i.e. the 'best' positions of the ligand when bound to the protein target. Moreover, *METADOCK* 2 performs a *BD*—i.e. it processes the whole protein surface in order to identify new allosteric binding, while most docking methods focus only on a user-specified part of the protein target.

2.1 The parameterized optimization procedure

The optimization procedure of *METADOCK* 2 is based on a parameterized metaheuristic scheme (Imbernón *et al.*, 2017). Metaheuristics are widely used to solve challenging optimization problems (a.k.a. NP-hard problems) (Rozenberg *et al.*, 2011) since they can provide a good solution in a minimal amount of time when there is not enough information or the computation is limited (Bianchi *et al.*, 2009). To do so, they only focus on the most promising solution candidates instead of exploring all possible solutions. This means they cannot guarantee finding the optimal solution albeit something close to it. There are many metaheuristics in the literature (Sörensen, 2015), such as *Distributed metaheuristics* (e.g. Scatter Search, Genetic Algorithms Ant Colony Optimization and Particle Swarm Optimization) and *Neighborhood metaheuristics* (e.g. Tabu Search, Hill Climbing, Simulated Annealing, etc.) (Llanes *et al.*, 2016). Finding out the 'best' metaheuristic to solve a particular task is not straightforward. Indeed, it is an optimization problem itself that requires, first of all, the most appropriate metaheuristic to be found for the targeted problem and also to tune the configuration parameters of the selected metaheuristic, which may increase the computational cost.

METADOCK 2 is based on a metaheuristic schema, which is able to generate a wide branch of metaheuristics. Algorithm 1 shows the sequential baseline of *METADOCK* 2. It is composed of several functions that are shared by many metaheuristics in the literature (Rozenberg *et al.*, 2011). These functions accept several input parameters (see Supplementary Table S1) to configure their internal procedure and to provide different functionality. Once the input parameters are established, *METADOCK* 2 applies the optimization procedure to obtain a subset of poses that minimize the scoring function.

Algorithm 1 shows that the optimization method starts by generating an initial population (S) of conformations on each surface region (see line 1) of the protein target. We can consider either just the enzyme active site (classical focused docking) or the exploration of the whole protein surface [BD (Sánchez-Linares *et al.*, 2012)]. Each initial population S is composed of a set of randomly generated poses (or individuals) with internal flexibility. At the initialization stage, *METADOCK* 2 can make improvements to some (or all) individuals. In addition, it is able to select which individuals are eventually included in the initial set to avoid elitist solutions. All of these features are established depending on the input parameters for this function (see *ParamIni* in Supplementary Table S1).

Next, the iterative procedure starts and proceeds until the end condition is met (lines 1–8). The *End_condition*(S, *ParamEnd*) function determines the stop criteria of *METADOCK* 2. Two different criteria are established, expressly, the maximum number of iterations and the number of iterations without improvement. The *Select*(S, *Ssel*, *ParamSel*) function chooses a percentage of conformations of the initial population (S) that will continue throughout the rest of the phases of the scheme. This function tries to maintain diversity by determining a set of the best and worst conformations. This selection is made based on the values of the scoring function

Algorithm 1 *METADOCK* 2 sequential baseline for the generation of diverse metaheuristics.

1: Initialize(*S*, ParamIni)
2: **while** not End_condition(*S*, ParamEnd) **do**
3: Select(*S*, Ssel, ParamSel)
4: Combine(Ssel, Scom, ParamCom)
5: Mutation(Scom, ParamMut)
6: Improve(Scom, ParamImp)
7: Include(Scom, *S*, ParamInc)
8: **end while**

for each individual. The *Combine*(*Ssel*, *Scom*, *ParamCom*) function mixes the selected conformations within the same region in pairs by combining (i) the best individuals among them, (ii) the worst individuals among them and finally (iii) the best individuals with the worst. Each pair of conformations will generate two new individuals with different orientations, but between both parents. The *Mutation*(*Scom*, *ParamMut*) is a new function added to *METADOCK* 2, whose main goal is to maintain the diversity of the combined set. Thus, *METADOCK* 2 is able to generate genetic-like algorithms by performing minor changes to the individuals previously combined. These changes include modifications to the spatial coordinates of the conformation, its orientation, or one of the rotatable links. The parameters associated with this function establish the percentage of conformations to be mutated and the intensity of this mutation. The *Improve*(*Scom*, *ParamImp*) function applies a local search within the neighborhood of the combined population. Finally, the *METADOCK* 2 procedure ends by including a subset of the conformations previously generated [*Include*(*Scom*, *S*, *ParamInc*)] in the next population to be considered.

2.2 The scoring functions under study

METADOCK 2 implements two different *scoring functions*. Both of them are based on traditional force field models and they take into account interaction in terms of dispersion–repulsion, hydrogen-bonds, electrostatics (ES) and desolvation, but their mathematical models introduce some particularities to their implementation. Two are the SF implementations and are of particular interest to us. First, we consider the SF of BINDSURF (Sánchez-Linares *et al.*, 2012) since it was one of the first GPU-based docking methods in the literature. Second, the implementation from AutoDock 4 is also chosen (Morris *et al.*, 1998) because it is one of the most widely used docking packages. From now on, we will refer to SF1 as the scoring function based on BINDSURF and SF2 as the scoring function used in AutoDock 4. Additionally, all terms used in SF1 and SF2 are explained below.

2.2.1 ES interactions

A molecule is composed of a set of atoms that form bonds with each other, implying a set of ES interactions between them. Equations (1) and (2) describe the mathematical models on which the ES interactions used in the SF1 and SF2 functions are based on, respectively. In those equations, n and m refer to the number of atoms of the ligand and the protein

$$\sum_{i=0}^{n}\sum_{j=0}^{m} k\left(\frac{q_i\ q_j}{r_{ij}}\right) \tag{1}$$

$$\sum_{i=0}^{n}\sum_{j=0}^{m} k\left(\frac{q_i\ q_j}{\epsilon_{r_{i,j}}\ r_{ij}}\right). \tag{2}$$

If the atoms are i (ligand) and j (receptor), the values q_i and q_j are the charges of the atoms i and j, r_{ij} is the atomic distance between both and k is the permittivity of a vacuum. Equation (2) shows the ES potential of AutoDock 4, whose main peculiarity is that it adds a new term to the denominator that multiplies the atomic distance. This term ($\epsilon_{r_{i,j}}$) can be interpreted as the dielectric constant of the medium, and it is mathematically approximated by the Mehler–Solmajer model (Mehler and Solmajer, 1991).

2.2.2 Van der Waals interactions

$$\sum_{i=0}^{n}\sum_{j=0}^{m} 4\epsilon_{ij}\left(\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right) \tag{3}$$

$$\sum_{i=0}^{n}\sum_{j=0}^{m}\left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}}\right). \tag{4}$$

This term combines the forces of attraction and repulsion between non-bonded atoms. Attractive forces are brought about by the distribution of charges and repulsive or short-range forces are created by Pauli's repulsion. SF1 implements Equation (3), where the depth of the potential is represented by ϵ_{ij} and σ_{ij} determines the equilibrium distance between the two atoms. The indexes n and m are the number of atoms of the ligand and the protein, respectively, and r_{ij} the atomic distance between atom i from the ligand and atom j from the receptor. Equation (4) shows the mathematical model of the implementation in SF2. The main difference between them is the way that coefficients A_{ij} and B_{ij} are calculated.

2.2.3 Hydrogen bond interactions

A hydrogen bond is a strong type of dipole–dipole directional ES interaction (Desiraju and Steiner, 1999). Equation (5) shows the mathematical model used by SF1 for this term

$$\sum_{i=0}^{n}\sum_{j=0}^{m}\cos\theta_{ij}\left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}}\right) + \sin\theta_{ij}\ 4\epsilon_{ij}\left(\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right) \tag{5}$$

where r_{ij} is the atomic distance and θ_{ij} is the angle that forms the atoms i and j.

Equation (6) shows the mathematical model implemented in SF2

$$\sum_{i=0}^{n}\sum_{j=0}^{m} E(t)\left(\frac{C'_{ij}}{r_{ij}^{12}} - \frac{D'_{ij}}{r_{ij}^{10}} + E_{hbond}\right). \tag{6}$$

The term $E(t)$ can be approximated to $\cos\theta_{ij}$ and the term E_{hbond} to $\sin\theta_{ij}$.

2.2.4 Desolvation potential

This term is only implemented in SF2, and it is based on the work of Wesson and Eisenberg (Eisenberg and McLachlan, 1986). This calculation is based on the absolute value of individual atomic charge, the type of atom, its volume and its associated solvation parameter. Equation (7) shows the term $S_i = a_i + k\ |q_i|$, where a_i is the solvation parameter for atom i, q_i is its partial charge and k is a Gaussian constant. S_j is similar to S_i for atom j

$$\sum_{i=0}^{n}\sum_{j=0}^{m}(S_i V_j + S_j V_i)e^{-r_{ij}/2\sigma^2}. \tag{7}$$

2.2.5 Ligand flexibility

The internal degrees of freedom of the ligand in terms of rotatable bonds are also considered in both scoring functions. The rotatable bonds are obtained by using OpenBabel (O'Boyle *et al.*, 2011).

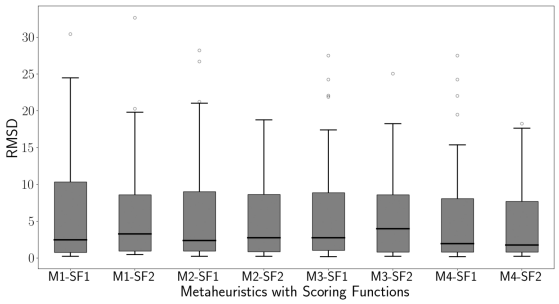

Fig. 1. Average RMSD values for redocking calculations (DUD dataset) using *METADOCK* 2 with four different metaheuristic configurations (*M*1, *M*2, *M*3 and *M*4) and two scoring functions (SF1 and SF2)

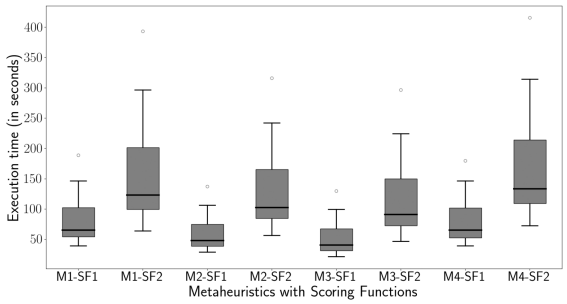

Fig. 2. Execution time in seconds for redocking calculations (DUD dataset) using *METADOCK* 2 with four different metaheuristic configurations (*M*1, *M*2, *M*3 and *M*4) and two scoring functions (SF1 and SF2)

2.3 Environmental setup

2.3.1 Testbed

Experiments were carried out on a heterogeneous computing system based on a multicore processor and a GPU. Specifically, the system has 12 Intel Xeon E5-2603 processors running at 1.7 GHz and plugged into a quad-channel motherboard endowed with 128 GB of DDR3 memory. It has a GPU NVIDIA Tesla Kepler K40c with 2880 CUDA cores (15 streaming multiprocessors and 192 GPU cores per multiprocessor) running at a boost clock of 0.88 GHz, giving a raw processing power of up to 5068 GFLOPS. The memory size is 12 GB of GDDR5. For compilation on the CPU, we used gcc 5.4.0 with the -O3 flag, and the CUDA toolkit version 8.0 was used for compilation on the GPU.

2.3.2 Schema configuration

METADOCK 2 has several input parameters that define the optimization method to be applied in the optimization process. Four different configurations have been used in this work (referred to as *M*1, *M*2, *M*3 and *M*4 in Supplementary Table S2). *M*1 is a hybrid-metaheuristic, which is close to a genetic algorithm. We set the initial population of *M*1 at 2048 individuals for each spot. The best elements and half of the resulting elements are mutated before the combination stage. This metaheuristic does not include a local search to improve the conformations. The second (*M*2) and third (*M*3) metaheuristics are based on an *evolutionary method* based on a *Scatter Search* algorithm with a population of 512 individuals. More specifically, in the case of *M*3, all chosen elements after the initial generation are combined and a local search in the neighborhood of each element is applied to obtain better solutions. The last metaheuristic (*M*4) is a method of intensive search in the neighborhood with a Monte Carlo technique. Once the search is done, a small percentage of the individuals in the initial phase are submitted for combination, mutation and improvement. Notice that we have used different population sizes for the metaheuristics because we are interested in studying the quality of prediction based on the variation of metaheuristic parameters. All individuals are considered for selection and combination in the algorithm.

2.4 Benchmarks

2.4.1 DUD dataset

The DUD (Mysinger *et al.*, 2012) is a publicly available dataset of about 100 000 ligands distributed over 40 protein targets. It includes structural information about active and decoy ligands for each target. The decoys are chosen as such because they are physicochemically similar to the actives but topologically different. We believe that the DUD provides a suitable benchmark with which to assess docking accuracy. In our work, we have performed redocking of active ligands and measured root-mean-square deviation (RMSD) and running times.

2.4.2 Flexible peptides

Peptide molecules are usually much bigger than small drug molecules that follow Lipinski rules and the ones that have been approved by FDA. Therefore, they represent a challenge for most docking methods, especially when we consider them as being flexible due to the high number of rotatable bonds. As a result, we have selected peptides from the LEADS-PEP dataset (Hauser and Windshügel, 2016), which was conveniently designed to this end.

2.4.3 Very large protein system

In order to test the BD approach implemented in *METADOCK* 2 against Autodock 4 and Autodock Vina, we looked for the largest protein–ligand systems in the PDB database and we selected the structure of yeast CP in complex with Belactosin C (PDB:3TDD), which contains around 100 000 atoms.

3 Results and discussion

3.1 Virtual screening accuracy and performance results from the docking methodologies for the DUD dataset

Docking simulations are performed on the 40 protein–ligand crystals from the DUD database. Figures 1 and 2 show the RMSD values and the execution times, respectively, for the *M*1, *M*2, *M*3 and *M*4 configurations of *METADOCK* 2 described in Section 2.3. These configurations are executed by using the SF1 and SF2 scoring functions (further details in Section 2.2).

Figure 1 shows that, on average, the lowest RMSD values are obtained with the combination *M*4-SF2. This can be explained as the *M*4 algorithm is based on an intensive neighborhood search where a small percentage of individuals are chosen for combination, mutation and improvement. Furthermore, the SF2 is the AutoDock 4-based scoring function which is more accurate than SF1. The reason is that SF1 introduces a penalty of between 0.1 and 0.7 Å. The differences between metaheuristics are even higher, reaching up to RMSD values of 1.52 Å, which places the search algorithm as the determining factor.

Regarding performance numbers, Figure 2 shows that *M*4-SF2, which is the most accurate configuration, is the slowest configuration in *METADOCK* 2. This implies a direct relationship between quality and performance, which allows the users of *METADOCK* 2 to benefit from some flexibility governed by this relationship, depending on the available hardware resources. It is also worth noting the differences between *M*4-SF1 and *M*4-SF2 in both performance and accuracy. The RMSD difference between them is 0.76 Å but the execution time is increased by a factor of 2.24×.

Supplementary Table S5 shows a comparison in terms of the RMSD and execution time in seconds among AutoDock 4, AutoDock Vina and the best configuration of *METADOCK* 2 for each DUD target. As for the *METADOCK* 2 numbers, we show the lowest RMSD value for each compound and its corresponding execution time and vice versa. As can be seen, on average, the best RMSD obtained by *METADOCK* 2 is also the best RMSD obtained

Table 1. BD results obtained with AutoDock 4, AutoDock Vina and *METADOCK* 2 when processing large systems (described in Sections 2.4.2 and 2.4.3)

PDB code	AutoDock 4	AutoDock Vina	*METADOCK* 2	
	RMSD	RMSD	RMSD SF1	RMSD SF2
2B6N	n.a.	n.a.	**7.66**	8.06
4BTB	18.2	13.81	**13.83**	20.38
2OXW	n.a.	n.a.	6.60	**5.78**
2OY2	n.a.	n.a.	**5.87**	6.18
1UOP	22.2	7.57	**3.84**	22.06
4C2C	n.a.	4.59	**1.59**	1.72
3BS4	17.2	2.79	**0.88**	1.73
3GQ1	n.a.	4.97	**1.82**	3.92
1TW6	4.7	4.52	**5.19**	7.99
4J44	n.a.	n.a.	**2.23**	15.12
1NVR	13.6	11.58	22.69	**4.78**
1B9J	22.8	3.63	**0.99**	1.55
3TDD	n.a.	n.a.	4.25	**2.37**
Median value	17.7	4.78	4.25	5.78

Note: Prediction accuracy is reported in terms of RMSD in Angstroms. For some instances, 'n.a.' stands for 'not available' since results cannot be obtained for that particular method. The minimum value obtained by METADOCK is highlighted in bold.

by all the docking methods under study. *METADOCK* 2 outperforms AutoDock 4 by a wide margin and slightly surpasses AutoDock Vina. The average execution time to reach this optimal value obtained by *METADOCK* 2 is a bit longer than the execution time for the multi-threaded version of Vina (58.5 versus 109.2 s). At this point, let us remind the reader that AutoDock Vina is executed using the OpenMP library on a 12-core multicore system. For a fair comparison, we decided to execute *METADOCK* 2 in its counterpart GPU version (i.e. a GPU Tesla K40c). However, *METADOCK* 2 could be efficiently run on a cluster with multiple GPUs, showing almost linear speed-up with the number of GPUs [we refer the reader to Imbernón *et al.* (2017) for more detailed information].

3.2 BD simulations on complex and large systems

Table 1 shows the accuracy prediction of performing BD simulations with AutoDock 4, AutoDock Vina and *METADOCK* 2 using SF1 and SF2. The second and third columns show the RMSD in Autodock 4 and in Autodock Vina, and in columns 4 and 5 the RMSD with the 2 METADOCK scoring functions (SF1 and SF2). Finally, in the sixth column called OPTIMAL SF, the best RMSD from the two METADOCK scoring functions is shown. Several complex systems are randomly selected from the PDB, including protein-flexible-peptide complexes, the LEADS-PEP dataset and large systems, such as 4TDD. It can be observed that in some cases AutoDock 4 and AutoDock Vina are not able to perform the BD simulations, while *METADOCK* 2 does. This might be due to the greater ability of *METADOCK* 2 to explore the optimization space in more detail. In addition, the general trend is that *METADOCK* 2 can obtain lower RMSD values for the same scoring function (except in the case of 1TW6), and that on average it obtains a much lower RMSD value.

4 Conclusions

This article introduces a novel method of molecular docking called *METADOCK* 2, which incorporates several features that make it a good candidate for performing docking simulations, especially for complex systems. Furthermore, it speeds up the whole simulation process with improved accuracy. The computational horsepower and algorithmics of *METADOCK* 2 allow the simulation of several complex systems that are not accessible with other docking methods, such as AutoDock 4 and Vina. In addition, the predictive precision obtained with *METADOCK* 2 outperforms both AutoDock 4 and Vina, providing a framework in which the user can select between several metaheuristics depending on the computational resources available.

METADOCK 2 is a framework that requires an initial configuration by setting metaheuristics related parameters. This configuration establishes the search method, which has been demonstrated as a key factor for getting accurate predictions. The results shown here are based solely on four fixed configurations that we have selected according to our previous knowledge. But we believe that other configurations may improve both performance and accuracy. This opens up new research paths to design and develop machine-learning-based expert systems that combine the latest advances in deep learning with metaheuristics to determine which the best parameter configuration is for each system. In addition, we are also working on finding an alternative scoring function and then parameterizing it to increase the accuracy of predictions.

Funding

This work was partially supported by the Fundación Séneca del Centro de Coordinación de la Investigación de la Región de Murcia [Projects 20813/PI/18, 20988/PI/18, 20524/PDC/18] and by the Spanish Ministry of Science, Innovation and Universities [TIN2016-78799-P (AEI/FEDER, UE), CTQ2017-87974-R]. The authors thankfully acknowledge the computer resources at CTE-POWER and the technical support provided by Barcelona Supercomputing Center - Centro Nacional de Supercomputación [RES-BCV-2018-3-0008].

Conflict of Interest: none declared.

References

Bianchi,L. *et al.* (2009) A survey on metaheuristics for stochastic combinatorial optimization. *Nat. Comput.*, **8**, 239–287.

Blum,C. and Roli,A. (2003) Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Comput. Surv.*, **35**, 268–308.

Cecilia,J.M. *et al.* (2018) High-throughput ant colony optimization on graphics processing units. *J. Parallel Distr. Com.*, **113**, 261–274.

Dakkak,A. *et al.* (2018) Accelerating reduction and scan using tensor core units. In: *Proceedings ACM International Conference on Supercomputing (ICS)*, 46–57, 2019.

Desiraju,G.R. and Steiner,T. (1999) *The Weak Hydrogen Bond: In Structural Chemistry and Biology* (International Union of Crystallography, Monographs on Crystallography, 9). Oxford University Press, Oxford and New York. ISBN: 0-19-850252-4.

Eisenberg,D. and McLachlan,A.D. (1986) Solvation energy in protein folding and binding. *Nature*, **319**, 199–203.

Ewing,T.J.A. *et al.* (2001) DOCK 4.0: search strategies for automated molecular docking of flexible molecule databases. *J. Comput. Aided Mol. Des.*, **15**, 411–428.

Friesner,R.A. *et al.* (2004) Glide: a new approach for rapid, accurate docking and scoring: method and assessment of docking accuracy. *J. Med. Chem.*, **47**, 1739–1749.

Guerrero,G.D. *et al.* (2014) A performance/cost evaluation for a GPU-based drug discovery application on volunteer computing. *BioMed Res. Int.*, **2014**, 1–8.

Hauser,A.S. and Windshügel,B. (2016) LEADS-PEP: a benchmark data set for assessment of peptide docking performance. *J. Chem. Inf. Model.*, **56**, 188–200.

Imbernón,B. *et al.* (2017) METADOCK: a parallel metaheuristic schema for virtual screening methods. *Int. J. High Perform. Comput. Appl.*, **32**, 1–15.

Llanes,A. *et al.* (2016) Soft computing techniques for the protein folding problem on high performance computing architectures. *Curr. Drug Targets*, **17**, 1626–1648.

McIntosh-Smith,S. *et al.* (2015) High performance in silico virtual drug screening on many-core processors. *Int. J. High Perform. Comput. Appl.*, **29**, 119–134.

Mehler,E.L. and Solmajer,T. (1991) Electrostatic effects in proteins: comparison of dielectric and charge models. *Protein Eng. Des. Sel.*, **4**, 903–910.

Morris,G.M. *et al*. (1998) Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. *J. Comput. Chem*., **19**, 1639–1662.

Mysinger,M.M. *et al*. (2012) Directory of useful decoys, enhanced (dud-e): better ligands and decoys for better benchmarking. *J. Med. Chem*., **55**, 6582–6594.

O'Boyle,N.M. *et al*. (2011) Open Babel: an open chemical toolbox. *J. Cheminform*., **3**, 33.

Rozenberg,G. *et al*. (2011) *Handbook of Natural Computing*. Springer.

Sakurai,Y. *et al*. (2015) Two-pore channels control Ebola virus host cell entry and are drug targets for disease treatment. *Science*, **347**, 995–998.

Sánchez-Linares,I. *et al*. (2012) High-throughput parallel blind virtual screening using BINDSURF. *BMC Bioinformatics*, **13**, S13.

Sliwoski,G. *et al*. (2014) Computational methods in drug discovery. *Pharmacol. Rev*., **66**, 334–395.

Sörensen,K. (2015) Metaheuristics the metaphor exposed. *Int. Trans. Oper. Res*., **22**, 3–18.

Yuan,S. *et al*. (2017) Structure-based discovery of clinically approved drugs as Zika virus NS2B-NS3 protease inhibitors that potently inhibit Zika virus infection in vitro and in vivo. *Antiviral Res*., **145**, 33–43.