A MEMETIC ALGORITHM ENABLES GLOBAL ALL-ATOM PROTEIN-PROTEIN DOCKING WITH SIDECHAIN FLEXIBILITY

Daniel Varela
Biochemistry and Structural Biology
Lund University
Lund, Sweden
daniel.varela@biochemistry.lu.se

Ingemar André
Biochemistry and Structural Biology
Lund University
Lund, Sweden
ingemar.andre@biochemistry.lu.se

April 1, 2021

ABSTRACT

Protein-protein docking plays a central role in the characterization and discovery of protein interactions in the cell. Complex formation is encoded by specific interactions at the atomic scale, but the computational cost of modeling proteins at this level often requires the use of simplified energy models, coarse-grained protein descriptions and rigid-body approximations. In this study we present EvoDOCK, which is an evolutionary-based docking algorithm that enables the identification of optimal docking orientations using an atomistic energy function and sidechain flexibility, employing a global search without prior information of the binding site. EvoDOCK is a memetic algorithm that combines the strength of a differential evolution algorithm for efficient exploration of the global search space with the benefits of a local optimization method, built on the Monte Carlo-based RosettaDOCK program, to optimize detailed atomic interactions. This approach resulted in substantial improvements in both sampling efficiency and computation speed compared to calculations using the local optimization method RosettaDOCK alone, with up to 35 times of reduction in computational cost. For all the ten systems investigated in this study, a highly accurate docking prediction could be identified as the lowest energy model with high efficiency. While protein-protein docking with EvoDOCK is still computationally expensive compared to many methods based on Fast Fourier Transforms (FFT), the results demonstrate the tractability of global docking proteins using an atomistic energy function while exploring sidechain flexibility. Comparison with FFT global docking demonstrated the benefits of using an all-atom energy function to identify native-like predictions. The sampling strategy in EvoDOCK can readily be tailored to include backbone flexibility in the search, which is often necessary to tackle more challenging docking challenges.

Keywords protein docking · Protein–protein complexes · memetic algorithm · global docking · sidechain flexibility

Introduction

Interactions of proteins are critical for the function of most cellular processes [1]. While the number of experimentally determined protein complexes is steadily increasing in the Protein Data Bank (PDB) [2], it still covers only a sliver of the interactome. Prediction of the structure of protein complexes and assemblies, therefore, continues to be a central problem in structural biology [3]. An explosion of protein sequence data coupled with improvements in machine learning has led to the extremely impressive performance of methods to predict individual protein structures [4]. The same advances have not been seen in the area of protein-protein docking, which remains a highly challenging problem. Several fundamental hurdles need to be overcome to achieve a significant improvement of the protein-protein docking methods. One is related to the challenge of accurately modeling the physical interactions between binding partners. A second involves difficulties in modeling changes in conformation that often accompany the formation of protein complexes. A third is related to the efficient exploration of the large conformational search space inherent to protein-protein docking. This study focuses on addressing the last challenge.

In many real applications of protein-protein docking, the starting point is typically the structure (models or experimental structures) of the individual interaction partners with an unknown binding mode. This is referred to as global docking [5]. Identifying the correct binding mode involves a six-dimensional search of rigid-body orientations, where three degrees of freedom correspond to rotations and three encode the relative translation between the binding partners. The flexibility of sidechains and backbone can further expand the search space. To reduce the computational cost, protein-protein docking methods typically use a rapid, but less accurate, search strategy to reduce the number of docking orientations that are then evaluated and refined by more atomistic approaches. Rigid body docking methods using Fast Fourier Transform (FFT) methods can very fast and, additionally, improvements of energy models that can be evaluated on a grid have enhanced their ability to identify good docking orientations [6]. An alternative strategy involves the use of coarse-grained models coupled to search strategies such as Metropolis Monte Carlo (MMC) [7] or evolutionary-based optimization methods [8]. The pool of docking orientations is then typically optimized with a refinement step, which couples the use of all-atom energy functions or force-fields with sidechain (and sometimes backbone) flexibility [9].

The use of simplified energy representations has some benefits beyond computational speed. In particular, they are less sensitive to the approximation of a rigid backbone often used in protein-protein docking. On the other hand, their use can also lead to the exploration of docking orientations that are far from the native state. The true energy landscape is typically better represented with all-atom energy functions. However, skipping the initial global search step with FFT/coarse-graining methods and directly optimizing docking orientations based on an all-atom energy function have not been feasible due to the high computational cost. Nonetheless, we show here that such an approach can be feasible if a more efficient conformational search method is employed. By combining an evolutionary algorithm (EA) for global search and a local search method to refine solutions, we demonstrate how both accurate and relatively fast atomistic protein-protein docking can be carried out.

One of the main advantages of EAs is their ability to efficiently explore large areas of the global search space compared to a local search strategy like Monte Carlo. Local methods identify a new solution based on a small change of the current solution (a strategy known as “exploitation”). On the other hand, population-based search methods, such as evolutionary algorithms, maintain many feasible solutions during each search iteration or generation. This allows a global search in different areas of the space of solutions, with the possibility of ”exploration” of new search areas through the genetic operators of the corresponding EA.

Both EAs and the local search can be combined in hybrid approaches known as “Memetic Algorithms” (MA), which aims to combine the exploration capability of an EA to identify the most promising regions of the search space while using local search to exploit the surroundings of the individual solutions [10]. In recent years, MAs has been successfully applied to solve complex optimization problems in many scientific domains [11][12] [13][14][15] and been demonstrated to be a high-efficiency strategy to explore fitness landscapes ([16][17]).

In this study, we present a method that uses the Differential Evolution (DE) algorithm as the evolutionary optimization framework and a MC-based docking approach as the local search method. DE [18] [19] is a population-based search algorithm that has been developed for optimization problems where possible solutions are encoded as real-valued vectors. This is the case in protein-protein docking where the relative position of two binding partners can be described by three translational and three rotational degrees of freedom. By converting DE into a memetic algorithm and combining it with a local optimization method, the search efficiency can be increased by refining the solutions proposed by the DE framework.

Several studies have presented evolutionary algorithms (such as genetic algorithms, swarm optimization and DE) to solve protein docking problems. Such approaches have been particularly employed for protein-ligand docking [20][21][22][23]. This includes some methods based on memetic algorithms, such as combinations of EAs with a variety of local search methods [24][25][26][27][28]. In the context of protein-protein docking, evolutionary algorithms have been used for rigid-body docking [29][30], while memetic approaches have also been defined [31][8]. Evolutionary algorithms can also enable exploration with backbone diversity. For example, SwarmDOCK is a memetic algorithm based on swarm optimization, that includes conformational variability through normal mode analysis [8].

These current applications of EAs to protein-protein docking have been limited to rigid-body sampling without sidechain flexibility and typically involve the use of simplified energy models and coarse-grained descriptions of proteins. The introduction of all-atom energy functions and conformational flexibility results in very rugged energy landscapes, which requires highly efficient search strategies to explore. In this study we present the protein-protein docking program EvoDOCK, which is based on a memetic algorithm where a differential evolution strategy is coupled to a Monte Carlo-based local search method built on RosettaDOCK[32]. The results demonstrate that highly accurate protein-protein docking predictions can be achieved by global search directly using an all-atom energy function while simultaneously exploring sidechain flexibility.

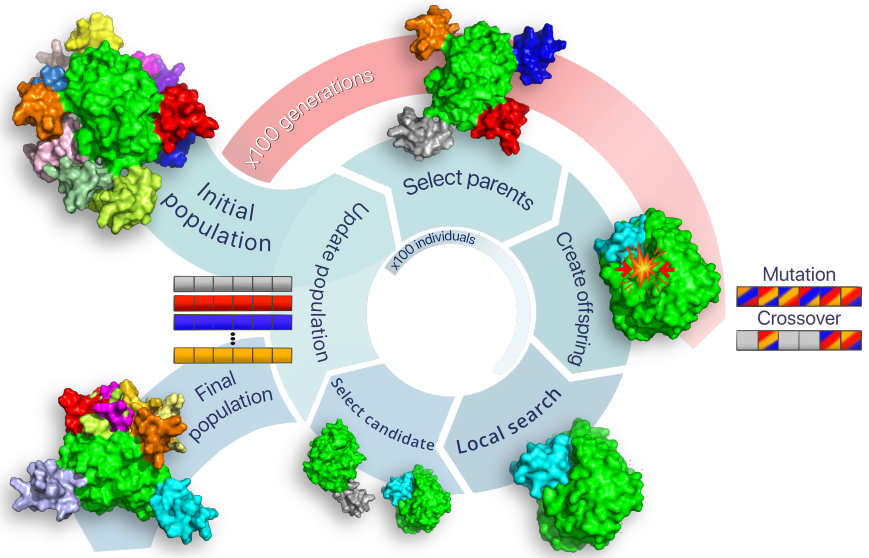


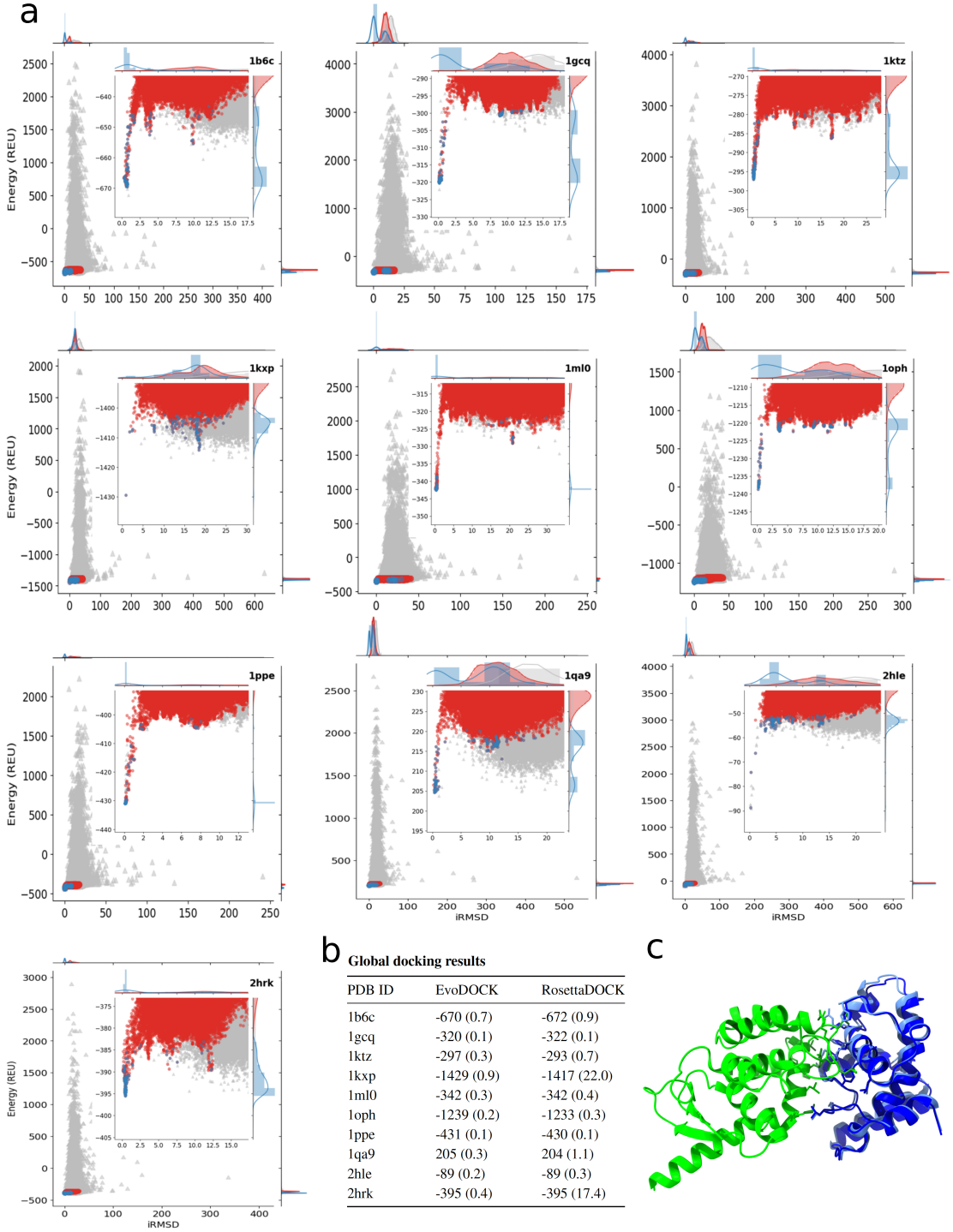
Figure 1: **Schematic representation of the EvoDOCK protein-protein docking methodology**. An initial population of 100 docking orientations is generated by randomly orienting binding partners around the green static protein. Out of the population, lowest energy individual and two parents are randomly selected to create a new offspring, illustrated here by the orange, blue and red members. The parents are combined to form the cyan offspring using mutation or crossover operations for each of the six docking degrees of freedom. This candidate typically has clashes that need to be resolved by a local search involving RosettaDock, which optimizes the rigid-body orientation and sidechain conformations. Two Monte Carlo cycles of RosettaDOCK is called in the default EvoDOCK protocol. The energy of the candidate is then compared to one of the selected patents (grey in this example) to determine whether it will replace it in the population. Each member of the population is compared with a candidate offspring in each generation. After 100 generations the lowest energy model is selected as the final prediction.

Results

The EvoDOCK methodology is schematically presented in Figure 1. An initial set of randomly generated docking orientations is prepared. This constitutes the initial population that is evolved over several generations. At each iteration of the Differential Evolution (DE) procedure, and for each current individual of the population, a trial solution is created by applying mutation and crossover operations on the parameters that describe the six docking degrees of freedom in the system based on the lowest energy individual. Each candidate is then optimized with a few cycles of a local refinement step in RosettaDOCK [32], which involves a Monte-Carlo based optimization of the rigid-body orientation, a combination of greedy and combinatorial rotamer optimization and rigid-body minimization. After this local search process, the lowest energy solutions are selected to continue to the next iteration. After the final iteration, the model with the lowest Rosetta energy is selected as the result of an EvoDOCK run.

Global docking performance of EvoDOCK

The performance of EvoDOCK was tested in global docking experiments in which the native binding mode is recovered without prior knowledge of the binding site and sidechain conformations in the binding interface. Since the focus here is on improvements in general search efficiency, we created a benchmark with complexes where the Rosetta all-atom energy function is able to identify the native as the energetically most favorable binding mode and where backbone adjustments are not required to identify the optimal solution (bound-bound docking). Based on the results in the RosettaDOCK local docking benchmarking study [32], we selected the 10 protein-protein complexes shown in Supplementary Figure 1. To compare the performance of EvoDOCK we also carried out global docking calculations using the standard rigid backbone docking protocol in RosettaDOCK. RosettaDOCK combines a Monte Carlo-based rigid body optimization in a coarse-grained representation followed by all-atom Monte Carlo optimization.



Global docking results

PDB ID	EvoDOCK	RosettaDOCK
1b6c	-670 (0.7)	-672 (0.9)
1gcq	-320 (0.1)	-322 (0.1)
1ktz	-297 (0.3)	-293 (0.7)
1kxp	-1429 (0.9)	-1417 (22.0)
1ml0	-342 (0.3)	-342 (0.4)
1oph	-1239 (0.2)	-1233 (0.3)
1ppe	-431 (0.1)	-430 (0.1)
1qa9	205 (0.3)	204 (1.1)
2hle	-89 (0.2)	-89 (0.3)
2hrk	-395 (0.4)	-395 (17.4)

Figure 2: **Distribution of the global docking results for 100,000 models**. **a**: Global docking results with EvoDOCK and RosettaDOCK. Each model is represented by a point where Rosetta full-atom Energy Units (REU) are shown on the y-axis with the corresponding interface RMSD (iRMSD) on the x-axis. The blue points are the result of 1,000 independent evolutionary processes by EvoDOCK, while the red points are the complete EvoDOCK population at the end of the trajectory(100.000 models). The RosettaDOCK models resulting from 100,000 independent trajectories. The inset shows a zoom into the region of low iRMSD to visualize the details of "near-native" decoys. **b**: Comparison of the lowest energy value obtained for each benchmark complex and its corresponding iRMSD (in parentheses) for EvoDOCK and RosettaDOCK. **c**: Lowest energy prediction for complex *2hrk*, with the crystal structure in light green and light blue and the EvoDOCK model in dark green and dark blue. **1qa9***: Complex *1qa9* shows a high energy value as a consequence of a backbone clash that is in the core of one of the binding partners.

The solution space accessible to EvoDOCK was comprehensively mapped by running 1,000 independent docking runs. This is considerably more than what is typically required to identify the native binding mode but enables the collection of appropriate statistics on docking performance. In total, a number of 100,000 (100x1000) members were evolved in these docking calculations for each complex. The same number of models was generated by RosettaDOCK for comparison with each complex. The docking energy landscape was summarized as the scatter plots in Figure 2, where the energy of each model was plotted against the interface RMSD (iRMSD) to the native complex. In all of the 10 cases, the 1,000 models predicted by EvoDOCK have low energy values and a significant fraction of models have low iRMSD values. Figure 2 also presents the complete evolved population for each benchmark complex (100 from each independent run). The range of iRMSD values sampled in this population are broader, but all members still have low energy values. In contrast, RosettaDOCK solutions have a wide range of energy and iRMSD values. High energy solutions are common and only a small fraction of models are typically sampled in the near-native area. For example, RosettaDOCK generates docking solutions with iRMSD values up to 500Å for the *1b6c* complex, and less than 2% of solutions have iRMSD values lower than 10 Å. The difference between the lowest and the highest energy values has a spread of more than 1,000 energy units. In contrast, EvoDOCK solutions converge to a narrow distribution near the native conformation. The maximum iRMSD value among the 100,000 models produced by EvoDOCK was 12 Å, with 94 % solutions having iRMSD values lower than 10 Å and the largest spread in energy less than 27 energy units. Similar results were observed for all the benchmark cases.

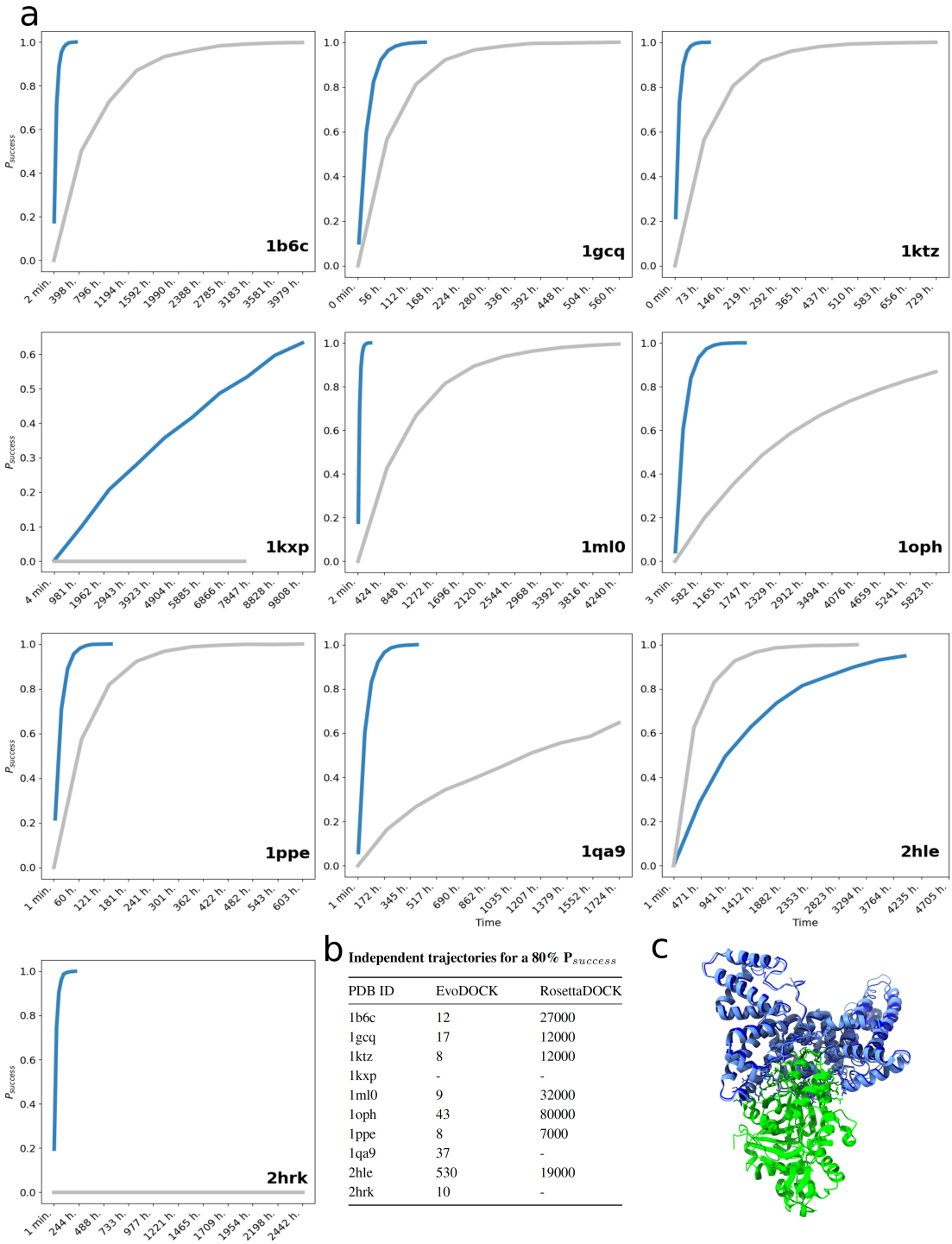
Focusing on a region near the native binding mode, we observe that EvoDOCK produces more models in this range than RosettaDOCK. In particular, below 2 Å iRMSD, EvoDOCK samples more models than RosettaDOCK in 9 out of 10 cases (Supplementary Table 1). With the exception of two complexes, 4 to 22% of all models predicted by EvoDOCK end up in this range, while RosettaDOCK never goes above 0.3%. In terms of the lowest energy identified in the simulations, the results are more similar. In 4 cases (*1ktz*, *1kxp*, *2hrk* and *1oph*) EvoDOCK produces the lowest energy model, RosettaDOCK finds the lowest energy in 3 cases (*1b6c*, *1gcq* and *1qa9*), while for the remaining 3 cases the energies are comparable.

In a blind docking experiment where the native structure is not known, solutions would be ranked by energy. Therefore, the quality of the docking results can also be assessed by comparing the iRMSD of the lowest energy solution. The table in Figure 2b summarizes these results. In all the cases, the lowest energy model obtained by EvoDOCK has a lower or equal iRMSD than RosettaDOCK. For two complexes, *1kxp* and *2hrk* (Figure 2c), RosettaDOCK is not able to predict the native binding mode. In the one case where RosettaDOCK produces a better prediction (*1gcq*), the difference to EvoDOCK is insignificant (0.03 Å).

Increase in computational efficiency of EvoDOCK

The results presented so far demonstrate that EvoDOCK performs a detailed sampling near the native binding mode. But an important characteristic of a docking method is also the computational cost. In a Monte Carlo-based docking algorithm, each run can be relatively short but it is necessary to sample a large number of different independent trajectories in order to sample near-native solutions. In contrast, an evolutionary algorithm optimizes several trajectories simultaneously and takes longer to complete, but with potentially fewer runs. To determine the number of EvoDOCK runs and aggregated simulation time necessary to achieve high-quality docking results, we have to define a metric for success. We considered a successful docking experiment one in which the lowest energy prediction has an iRMSD of 2.0 Å or less to the native binding mode, in which the molecular interactions are likely to be very similar to the native complex. Since docking simulations are stochastic, we need to define a probabilistic measure. $P_{success}$ measures the probability of success (iRMSD $\leq$ 2.0 Å for lowest energy prediction) with a given number of models generated. To evaluate $P_{success}$, we sampled data sets using a bootstrap method based on the 1,000 EvoDOCK runs (with a total of 100,000 models) for each complex, following the strategy described by Chaudhury et al. [32].

In Figure 3, $P_{success}$ is plotted against computational time (in hours) on a single core of an Intel Xeon E5-2650 v3 processor (2.3 GHz). For comparison we have evaluated the same metric on the RosettaDOCK simulations. As it can be seen in Supplementary Table 2, in 9 out of 10 cases, given the same $P_{success}$, EvoDOCK reduces the necessary computational time in comparison to RosettaDOCK. The docking results can be divided into three categories. In the first category, both EvoDOCK and RosettaDOCK successfully produce high-quality docking solutions and with $P_{success}$ values up to 100% (*1b6c*, *1gcq*, *1ktz*, *1ml0*, *1ppe*). For these cases, we can compare computational efficiency by comparing the computational time required to achieve a specific probability of success. In all these cases, EvoDOCK provides a significant speed boost from 4 to 38 times faster. A second category corresponds to the cases where EvoDOCK succeed in reaching $P_{success}$ values, while RosettaDOCK plateaus at lower values. For example, for *1qa9*, RosettaDOCK achieves a maximum $P_{success}$ of 65%, a level that EvoDOCK can reach more than 30 times faster. Similarly, the maximum $P_{success}$ for complex *1oph* for RosettaDOCK is near 90%, which can be reached 15 times faster by EvoDOCK. In the third category of results, RosettaDOCK cannot identify any near-native solutions



b **Independent trajectories for a 80% $P_{success}$**

PDB ID	EvoDOCK	RosettaDOCK
1b6c	12	27000
1gcq	17	12000
1ktz	8	12000
1kxp	-	-
1ml0	9	32000
1oph	43	80000
1ppe	8	7000
1qa9	37	-
2hle	530	19000
2hrk	10	-

c

Figure 3: **Computational efficiency comparison of EvoDOCK and RosettaDOCK.a**: Comparison of the efficiency of EvoDOCK and RosettaDOCK in identifying "successful" docking orientations. A prediction is defined as "successful" when the iRMSD of the lowest energy individual is below a threshold of 2.0 Å. Plots show the value of $P_{success}$ (y-axis), obtained by bootstrapping, for sampled sets of predictions as a function of computational time (x-axis). Computational time was calculated by taking the average time for completing one run/model and multiplying it with the total number of runs/trajectories for EvoDock/RosettaDOCK respectively. **b**: Number of independent runs necessary to achieve a $P_{success}$ of 80% with EvoDOCK and RosettaDOCK. Each EvoDOCK run involves a population of 100 models. **c**: Lowest energy prediction for complex *1xp*, with the crystal structure in light green and light blue and the EvoDOCK model in dark green and dark blue.

(*2hrk* and *1kxp*). This is probably due to enhanced sampling around false minima at high iRMSD, which can be seen in Figure 2. EvoDOCK finds near-native solutions, but achieves only a $P_{success}$ of 65% with *1kxp*, suggesting that this complex has a difficult landscape to explore. Finally, for one case, *2hle*, RosettaDOCK is more efficient at the sampling binding mode.

We can also quantify the number of runs needed by either method to achieve a $P_{success}$ value higher than 80%, see table in Figure 3b. EvoDOCK needs an average of 18 runs in order to reach the $P_{success}$ threshold value for 8 out of 9 cases this threshold is reached, with 100 solutions (population size) minimized simultaneously during each run. In comparison, RosettaDOCK requires an average of 29,500 independent Monte Carlo trajectories to reach the same threshold.

Comparison with FFT-based global docking

Global protein-protein docking is typically carried out by FFT-based methods like ZDOCK [33], GRAMM [34], pyDOCK [35], ftDOCK [36] and ClusPro [37] . We selected ClusPro to compare with EvoDOCK since its one of the most successful automatic methods in CAPRI over the last decade and is one of the most popular docking servers [38]. The FFT calculation in ClusPro is carried out by PIPER [39]. Low energy models are then clustered and the center of most populous clusters are then energy minimized using the CHARMM potential [40]. In the table in Figure 2b, the iRMSD of cluster center of the most populous cluster is presented using the 'Balanced' scoring scheme during the FFT. The values range is between 1.3 and 23.1 Å, with 6 out of 10 predictions having over 2 Å iRMSD. Results are similar with the other scoring schemes applied with the program. However, we can also look at iRMSD of all the models in all the populous clusters. The minimum iRMSD values vary between 1.1 and 2.4 Å in this set. This demonstrates that ClusPro samples low iRMSD binding modes, although the challenge is to pick them out based on the energy and clustering metrics used to rank models.

Finally, we compared the computational time with the FFT-based ClusPro. Since the timings were not available from the ClusPro server we ran docking calculations with FFT program PIPER, which is the most time-consuming part of ClusPro. The results are found in Supplementary Table 7 and show that PIPER needs an average computational time for four the four different scoring schemes used by ClusPro of 35 hours. On the other hand, EvoDOCK, to achieve a $P_{success}$ of 60% and 80%, requires a total computational time of 45 and 78 hours, respectively. That is a significant improvement compared to the more than 800/1000 hours of computational time that RosettaDOCK needs to achieve a $P_{success}$ of 60/80%, respectively.

Local Docking results

Local docking refinement serves as an important complement to global docking. It enables the improvement of solutions discovered in global docking simulations, the refinement of coarse-grained docking solutions, the characterization of the local energy landscape and the discovery of energy funnels. To evaluate EvoDOCK in the context of local docking refinement, the method was adapted to enable sampling around a given starting point, rather than from random positions as in the global docking scenario. In the local docking simulations, the initial positions are randomly perturbed around a given starting point by adding randomly selected rotation and translation perturbations. After this initiation, the methodology is the same as in the global docking case. Experiments were carried out using the same benchmark complexes used in the global docking, starting from the native binding mode and RosettaDOCK simulations were used as reference simulations using the same set of maximum perturbations. For EvoDOCK, we filtered out the starting states with less than 2Å to the native binding mode so that the initial population did not include the desired binding mode.

For both EvoDOCK and RosettaDOCK, local refinement results converge to near-native binding modes (Supplementary Figure 2). As can be observed in table Supplementary Table 3, both methods achieve a very similar result in terms of energies and iRMSD values, with dense sampling around the native binding mode. To map out the energy landscape around the native binding mode more comprehensively, the complete population obtained at the last iteration of the evolutionary process can be analyzed. This is demonstrated in Figure 4 for the complex *1ktz*. The additional models highlight alternative minima in the docking landscape not found in the energy-optimized EvoDOCK solutions. When all local refinement results are evaluated with this approach (Supplementary Figure 2) we observe that EvoDOCK is able to densely sample the local docking energy landscape.

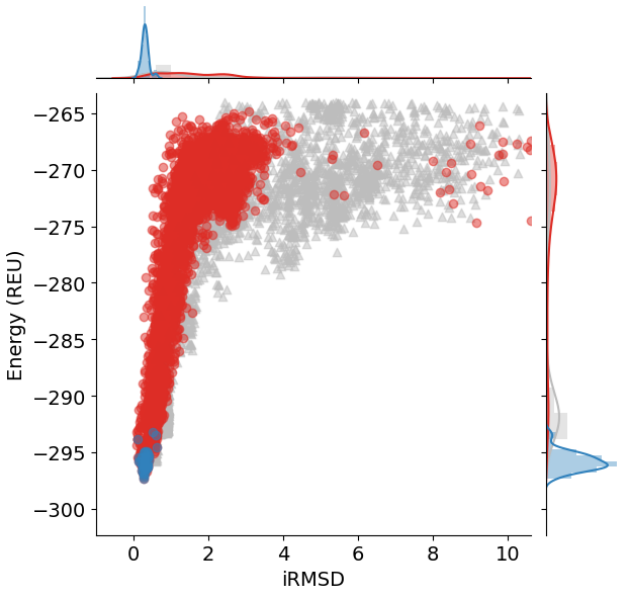


Figure 4: **Exploration of the energy landscape with the local docking for the benchmark complex *1ktz***. A plot of Rosetta energy (y-axis) versus interface RMSD (iRMSD, x-axis) for each sampled model. The 5,000 models obtained in the optimized population by EvoDOCK are shown in red (the lowest energy individual achieved by each of 50 runs is represented in blue) and the 5,000 RosettaDOCK models are shown in grey for comparison.

Discussion

This study presents an alternative sampling strategy to improve the efficiency of conformational search for the computationally challenging problem of protein-protein docking. Typically, docking approaches with sampling strategies based on local search, such as the MMC algorithm used by RosettaDOCK, reduce the computational cost by utilizing a coarse-grained phase before the all-atom refinement stage. In contrast, the methodology presented here involves global sampling directly with an all-atom representation. Although atomistic modeling is computationally expensive and the energy landscape is rugged and multimodal, the balance between exploration and exploitation provided by the combination of DE and the MMC refinement results in an efficient sampling of the docking landscapes. The combination provides a method that substantially improves on the local search strategy employed by MMC in terms of identifying native binding modes and significantly lowers the computational cost.

In contrast to the local search strategy provided by independent runs of MMC, DE is a population-based search method, which allows for information exchange between different solutions. As indicated by Feoktistov [41], the fundamental idea of the algorithm is that it intrinsically balances exploration and exploitation in the search by modifying solutions by vector differences between current solutions. At the early iterations, the exploration is large because individuals are very different from each other. As the evolution proceeds, the population is guided by the funnel-shape of the Rosetta energy landscape and converges towards a minimum, and the smaller differences between individuals lead to a stronger emphasis on exploitation which results in refinement of solutions.

The use of standard mutation and recombination operators, however, can lead to unfeasible candidate solutions with clashes in the backbone or sidechains. To address this issue, EvoDOCK uses a memetic strategy [10], where DE solutions are repaired by a local refinement method that resolves backbone clashes and optimizes sidechain conformations. As detailed in Supplementary Figure 3, this hybrid approach is essential for providing the evolutionary selection process with low energy candidates and in order to progressively optimize the solutions of the population. The default choice of two refinement steps per model provides a good balance between computational speed and accuracy. With a single refinement step, the prediction accuracy is somewhat reduced but is still comparable (Supplementary Table 4). This is associated with an average 20% reduction in computational time, but the decrease is not uniform across all the benchmark complexes, with some complexes taking longer time with one rather than two refinement steps (Supplementary Table 5). This suggests that the local refinement step also contributes to the convergence of the evolutionary algorithm. In the absence of the local refinement stage, the DE algorithm plateaus at high energy values (Supplementary Figure 4). Increasing the number of DE iterations in order to obtain the same computational cost as

with runs using the local refinement step cannot compensate (Supplementary Figure 5). Hence the combination of DE and local refinement by RosettaDOCK is critical for the performance of EvoDOCK.

Although EvoDOCK provides competitive results when finding the lowest iRMSD solution, it may not always provide the lowest energy solution. Nonetheless, DE can be tuned to improve the energy optimization or to show other desired traits. For benchmarking purposes, we used a single set of parameters for each studied complex. However, the behavior of the DE algorithm can readily be tailored with parameter changes or adaption of strategies to the benefit of each specific complex. For example, more diverse DE solutions can be produced with a variety of niching methods, as previously done with crowding in the context of protein structure prediction [42][43].

All-atom global docking with sidechain flexibility still comes with a significant computational cost, but the speed boost provided by EvoDOCK can enable exploration of single complexes without access to supercomputing resources, especially if a handful of CPU cores can be utilized. With four cores, the resolution of the docking of a protein complex would be done in an average of 18 runs, which can be completed in a day on current computer hardware. EvoDOCK is also more efficient in exploring local docking landscapes, so that additional binding modes sampled with faster global docking procedures like FFT can be refined with the same level of computational cost. Although not presented here, we also observed that optimizing the parameters for DE on a per system basis can lead to further improvement of sampling efficiency.

Even though EvoDOCK is substantially more efficient at exploring the global docking space than a standard MC-based method, it still comes with a substantial computational cost. FFT-based methods can be made blazingly fast, especially with GPU-enabled calculations. This speed often comes with a difficulty in ranking the models out of the prediction. As our example with ClusPro demonstrated, FFT can sample highly accurate binding modes but these can be difficult to identify with the coarse-grained FFT scoring functions and clustering schemes. Local docking refinement of the set of models is likely to identify the correct binding mode in our test cases and improve the prediction quality. With EvoDOCK a set of top FFT predictions could be used as an initial population to efficiently identify the native binding mode in a single run.

A central challenge in protein-protein docking is modeling of conformational changes upon binding and to utilize structures produced by comparative modeling with imperfect backbones [9]. The systems used in this study were selected to be solvable without the need for substantial backbone flexibility. Hence, fixed backbone EvoDOCK would not be expected to solve the most difficult protein-protein docking challenges. However, EvoDOCK can readily be tailored to introduce conformational variation. For example, the docking can be initialized with a population of pre-generated set of conformers. This would expand the computational cost, but would still be feasible with a limited number of conformers, especially in the local docking scenario. Alternatively, the local refinement state could include a backbone minimization step, as it has been demonstrated for RosettaDOCK with symmetrical complexes [44].

In summary, we demonstrate that accurate global and local docking with sidechain flexibility can be achieved with an all-atom energy function by a memetic algorithm. The relative simplicity and flexibility of the methodology makes it an appropriate base for the development of more complex docking protocols with a possibility of custom tailored protocols towards specific systems of interest.

Methods

Preparation of structures

The input complexes were prepared by the docking prepack protocol in Rosetta [32] to remove clashes between the sidechains in the experimental structure, using a command line found in the Supplementary Methods. The sidechain repacking is done when the binding partners are separated so that the sidechain conformations in the binding interface are not kept in the input structure for docking.

Positioning of binding partners in EvoDOCK

In protein-protein docking, one protein partner is normally fixed (known as the "static molecule") and the other protein partner moves in relation to the static protein (the so-called "moving molecule"). The relative orientation between the binding partners can be specified by a six-dimensional vector, where three values correspond to a rotation (ψ, θ, ϕ) and three to a translation (x, y, z). The rotation is specified by three Euler angles obtained by the orientation of the "moving molecule" with respect to a fixed coordinate system. The translation is defined in relation to the relative position of the "static molecule". Rotation values are sampled at a maximum range of $[-180, 180]$, while translation values during the evolutionary process are sampled in the following way:

1. The moving molecule is randomly placed with respect to the rigid molecule at a large "non-contact" distance.
2. The local search method calls a simple procedure to slide the moving molecule into atomic contact with the static molecule.
3. The ensemble is analyzed to identify the largest value of x, y, and z in the translation vector after the local search. These are used as the upper bound in the evolutionary process for the translation vector.

This procedure ensures that the initial solutions are close enough to the "static molecule" and that each x, y and z translation is constrained by a reasonable value.

The Memetic Differential Evolution

The algorithm begins by creating a random set of candidate solutions defined as *N*umber of *P*arents (NP). The six-dimensional vector containing the information about rotation and translation of each candidate is randomly sampled at the ranges described in the previous section. That is, the genotype of each solution corresponds to the six dimensional vector. Additionally, these values are normalized in the interval [-1,1] for use with the DE genetic operators, which is frequently used in classic implementations of EAs. When the fitness is assigned, the encoded parameters are decoded to their corresponding values.

The standard implementation of DE algorithm [19] is employed to optimize the solutions guided by a fitness function. In our docking calculations, the fitness function corresponds to the Rosetta full-atom energy function (*ref2015*). In order to create new candidate solutions, DE uses the mutation and crossover operators following the procedure defined in Algorithm 1. The method starts by selecting three different vectors (randomly chosen) from the current population. Then, for each of the 6 degrees of freedom (DoFs), given a crossover probability(*CR*), the candidate parameters are selected over a given vector of the population (target vector $P[i]$) and a "mutant vector". The mutation operation creates a difference from vectors ($P[i_2][j] - P[i_3][j]$) multiplied by factor (*F*) and then added to the base vector $P[i_1]$. The differential evolution strategy configuration used is *DE/bin/1/best*, where '*'bin"* indicates that binomial crossover is employed during the formation of the trial population and *"best"* that the lowest energy individual is used as the base vector for the mutation operation, while the remaining two vectors are selected randomly. The aforementioned F and CR parameters are used to control the search performance (the exploration and exploitation). During our experiments, we have used a fixed set configuration where $F = 0.9$ (usually between $[0, 2]$) and $CR = 0.3$ (usually in range $[0, 1]$), determined experimentally to obtain a balanced search between exploration and exploitation.

Algorithm 1 Create candidate O[i] from parent P[i]

```
randomly select parents P[i_1], P[i_2], P[i_3] where i_1 ≠ i_2 ≠ i_3 ≠ i
for j=0; j <DoFs; j++ do
    if (RANDOM(0,1) < CR) then
        mutation operation
        O[i][j] = P[i_i][j] + F(P[i_2][j] − P[i_3][j])
    else
        O[i][j] = P[i][j]
    end if
end for
```

In the memetic algorithm, DE is combined with a local search (LS) applied to every candidate, where the local search operator acts as a repair mechanism and a refinement method for the candidate. Further details of the effect of LS during the evolutionary process are shown in the Supplementary Methods, Figure 3. The local search operator starts with a *slide_into_contact* method provided by Rosetta, where the "moving molecule" is translated along a vector at the center of mass until it contacts the static body, followed by two cycles of a rigid-body MMC energy minimization and a combinatorial sidechain optimization. As in standard DE, the fitness of the final trial individual $O[i]$ is compared with the fitness of the corresponding target individual $P[i]$, to determine which one enters the population in the next generation.

The settings used to control the RosettaDOCK local search are shown in Supplementary Methods and a comparison between different settings (found in Supplementary Methods) has been analyzed in terms of energy sampling (Supplementary Table 4), computational time (Supplementary Table 5) and how the energy improvement at candidates achieved by the Local Search method contributes to the evolutionary process (Supplementary Figure 4) .

The EvoDOCK code can found in GitHub and it was implemented with Python, using PyRosetta [45] version 4. The EvoDOCK command line and configuration files are included in Supplementary Methods.

RosettaDOCK

Fixed-backbone RosettaDOCK models were obtained using the settings recommended in a previous benchmark study [32]. The method starts with randomization of the initial position by using the Rosetta parameters *randomize1*, *randomize2* and *spin*. The *nstruct* parameter, which defines the desired number of solutions to be obtained, was set to $100,000$ in order to generate the model dataset. For each solution, RosettaDOCK performs 500 MMC cycles using the coarse-grained representation and 50 MMC cycles with the full-atom representation (the local search algorithm used at EvoDOCK) but with a greedy sidechain optimization replacing the combinatorial step in 7 out of 8 cycles. The command line and further explanation parameters, used for RosettaDOCK can be found in Supplementary Methods.

Setup for local docking

Local docking with EvoDOCK follows the same approach used with global docking. Local docking assumes a known starting binding mode and constrains the sampling search around it. In local docking with RosettaDOCK a maximum rotation perturbation of 3 $^\circ$ and a maximum translation of 8 Å are typically used by using the flag *dock_pert 3 8*. The command line for the docking perturbation can be found in Supplementary Methods. Since EvoDOCK is a population-based search, the random population created using a local perturbation approach may include a nearby solution to the "native" complex. In order to avoid the fast convergence of the algorithm to that given solution, the initial solutions are sampled with the same maximum perturbations (3° for rotation and 8 Å for translation), followed by filtering the starting points to avoid models with an RMSD to the starting model of less than 2 Å). Both methods are used to find 5,000 optimized models, EvoDOCK by running 50 independent runs with 100 individuals, while RosettaDOCK "nstruct" parameter is set to 5,000.

Probabilistic measure for docking

For each target complex, we generated re-sampled decoy sets and calculated bootstrap statistical measures based on the observed docking results from each re-sampled decoy set. We generated B resampled decoy sets with replacement. From each N set of re-sampled decoy sets we calculated the probability of observing a successful docking result (lowest energy solution with iRMSD lower than 2 Å) and calculated the average for a number of re-sampled datasets of 5,000 ($B = 5000$) datasets as follows:

$$P_{success} = \frac{1}{B} \sum_{B}^{i=1} \{iRMSD(lowest_energy_solution(N_i)) \leq 2\} \tag{1}$$

The computational time of the sampled dataset was calculated by multiplying the number of decoys/runs with the average time expended to obtain an independent trajectory with the given method (RosettaDOCK or EvoDOCK). The time to obtain an average model for EvoDOCK and RosettaDOCK is found in Supplementary Methods, Table 6.

Code availability

EvoDOCK is available in GitHub https://github.com/Andre-lab/evodock.

Acknowledgments

This work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant agreement No. 771820). The computations were enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at HPC2N partially funded by the Swedish Research Council through grant agreement no. 2020/5-308 and the compute center LUNARC at Lund University with agreement no. 2021/2-59.

Supplementary Methods

Supplementary data are available online.

References

[1] B. Alberts, "The cell as a collection of protein machines: preparing the next generation of molecular biologists," *Cell*, vol. 92, no. 3, pp. 291–294, 1998.

[2] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The protein data bank," *Nucleic acids research*, vol. 28, no. 1, pp. 235–242, 2000.

[3] M. F. Lensink, S. Velankar, and S. J. Wodak, "Modeling protein–protein and protein–peptide complexes: Capri 6th edition," *Proteins: Structure, Function, and Bioinformatics*, vol. 85, no. 3, pp. 359–377, 2017.

[4] E. Callaway, "'It will change everything': DeepMind's AI makes gigantic leap in solving protein structures," *Nature*, vol. 588, no. 7837, pp. 203–204, 2020.

[5] S.-Y. Huang, "Exploring the potential of global protein–protein docking: an overview and critical assessment of current programs for automatic ab initio docking," *Drug Discovery Today*, vol. 20, no. 8, pp. 969–977, 2015.

[6] I. T. Desta, K. A. Porter, B. Xia, D. Kozakov, and S. Vajda, "Performance and Its Limits in Rigid Body Protein-Protein Docking," *Structure*, vol. 28, no. 9, pp. 1071–1081.e3, 2020.

[7] J. J. Gray, S. Moughon, C. Wang, O. Schueler-Furman, B. Kuhlman, C. A. Rohl, and D. Baker, "Protein–protein docking with simultaneous optimization of rigid-body displacement and side-chain conformations," *Journal of Molecular Biology*, vol. 331, no. 1, pp. 281–299, 2003.

[8] M. Torchala, T. Gerguri, R. A. Chaleil, P. Gordon, F. Russell, M. Keshani, and P. A. Bates, "Enhanced sampling of protein conformational states for dynamic cross-docking within the protein-protein docking server swarmdock," *Proteins: Structure, Function, and Bioinformatics*, vol. 88, no. 8, pp. 962–972, 2020.

[9] A. Harmalkar and J. J. Gray, "Advances to tackle backbone flexibility in protein docking," *Current Opinion in Structural Biology*, vol. 67, pp. 178–186, 2021.

[10] P. Moscato and C. Cotta, *A Gentle Introduction to Memetic Algorithms*, pp. 105–144. Boston, MA: Springer US, 2003.

[11] T. Rogalsky and R. Derksen, "Hybridization of differential evolution for aerodynamic design," in *Proceedings of the 8th annual conference of the computational fluid dynamics society of Canada*, pp. 729–736, Citeseer, 2000.

[12] V. Tirronen, F. Neri, T. Kärkkäinen, K. Majava, and T. Rossi, "An enhanced memetic differential evolution in filter design for defect detection in paper production," *Evolutionary Computation*, vol. 16, no. 4, pp. 529–555, 2008.

[13] Q. H. Nguyen, Y.-S. Ong, and M. H. Lim, "A probabilistic memetic framework," *IEEE Transactions on evolutionary Computation*, vol. 13, no. 3, pp. 604–623, 2009.

[14] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE transactions on evolutionary computation*, vol. 7, no. 2, pp. 204–223, 2003.

[15] K. C. Tan, S. C. Chiam, A. Mamun, and C. K. Goh, "Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization," *European Journal of Operational Research*, vol. 197, no. 2, pp. 701–713, 2009.

[16] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM computing surveys (CSUR)*, vol. 45, no. 3, pp. 1–33, 2013.

[17] A. Lala, V. Kolici, F. Xhafa, X. Herrero, and A. Barolli, "On local vs. population-based heuristics for ground station scheduling," in *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*, pp. 267–275, IEEE, 2015.

[18] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[19] K. V. Price, R. M. Storn, and J. A. Lampinen, "The differential evolution algorithm," *Differential evolution: a practical approach to global optimization*, pp. 37–134, 2005.

[20] G. Jones, P. Willett, R. C. Glen, A. R. Leach, and R. Taylor, "Development and validation of a genetic algorithm for flexible docking," *Journal of molecular biology*, vol. 267, no. 3, pp. 727–748, 1997.

[21] L. Kang, H. Li, H. Jiang, and X. Wang, "An improved adaptive genetic algorithm for protein–ligand docking," *Journal of computer-aided molecular design*, vol. 23, no. 1, pp. 1–12, 2009.

[22] C. S. de Magalhães, V. Dias, and L. E. Dardenne, "Comparison of differential evolution variants for the molecular ligand-receptor docking problem," in *2015 Latin America Congress on Computational Intelligence (LA-CCI)*, pp. 1–6, IEEE, 2015.

[23] P. H. Narloch and M. Dorn, "Rosetta ligand-protein docking with self-adaptive differential evolution," in *2019 IEEE 19th International Conference on Bioinformatics and Bioengineering (BIBE)*, pp. 23–30, IEEE, 2019.

[24] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, and A. J. Olson, "Automated docking using a lamarckian genetic algorithm and an empirical binding free energy function," *Journal of computational chemistry*, vol. 19, no. 14, pp. 1639–1662, 1998.

[25] R. Thomsen and M. H. Christensen, "Moldock: a new technique for high-accuracy molecular docking," *Journal of medicinal chemistry*, vol. 49, no. 11, pp. 3315–3321, 2006.

[26] H.-M. Chen, B.-F. Liu, H.-L. Huang, S.-F. Hwang, and S.-Y. Ho, "Sodock: Swarm optimization for highly flexible protein–ligand docking," *Journal of computational chemistry*, vol. 28, no. 2, pp. 612–623, 2007.

[27] A. Grosdidier, V. Zoete, and O. Michielin, "Blind docking of 260 protein–ligand complexes with eadock 2.0," *Journal of computational chemistry*, vol. 30, no. 13, pp. 2021–2030, 2009.

[28] H. W. Chung, S. Cho, K.-R. Lee, and K. Lee, "Self-adaptive differential evolution algorithm incorporating local search for protein-ligand docking," *Journal of Physics Conference Series*, vol. 410, pp. 2030–, 02 2013.

[29] E. J. Gardiner, P. Willett, and P. J. Artymiuk, "Protein docking using a genetic algorithm," *Proteins: Structure, Function, and Bioinformatics*, vol. 44, no. 1, pp. 44–56, 2001.

[30] S. Sudha, S. Baskar, and S. Krishnaswamy, "Protein docking using constrained self-adaptive differential evolution algorithm," *Soft Computing*, vol. 23, no. 22, pp. 11651–11669, 2019.

[31] P. F. Leonhart, P. H. Narloch, and M. Dorn, "A self-adaptive local search coordination in multimeme memetic algorithm for molecular docking," in *International Conference on Computational Science*, pp. 145–159, Springer, 2019.

[32] S. Chaudhury, M. Berrondo, B. D. Weitzner, P. Muthu, H. Bergman, and J. J. Gray, "Benchmarking and analysis of protein docking performance in rosetta v3. 2," *PloS One*, vol. 6, no. 8, p. e22477, 2011.

[33] R. Chen, L. Li, and Z. Weng, "Zdock: an initial-stage protein-docking algorithm," *Proteins: Structure, Function, and Bioinformatics*, vol. 52, no. 1, pp. 80–87, 2003.

[34] A. Tovchigrechko and I. A. Vakser, "Development and testing of an automated approach to protein docking," *Proteins: Structure, Function, and Bioinformatics*, vol. 60, no. 2, pp. 296–301, 2005.

[35] T. M.-K. Cheng, T. L. Blundell, and J. Fernandez-Recio, "pydock: Electrostatics and desolvation for effective scoring of rigid-body protein–protein docking," *Proteins: Structure, Function, and Bioinformatics*, vol. 68, no. 2, pp. 503–515, 2007.

[36] H. A. Gabb, R. M. Jackson, and M. J. Sternberg, "Modelling protein docking using shape complementarity, electrostatics and biochemical information," *Journal of molecular biology*, vol. 272, no. 1, pp. 106–120, 1997.

[37] D. Kozakov, D. R. Hall, B. Xia, K. A. Porter, D. Padhorny, C. Yueh, D. Beglov, and S. Vajda, "The cluspro web server for protein–protein docking," *Nature protocols*, vol. 12, no. 2, p. 255, 2017.

[38] I. T. Desta, K. A. Porter, B. Xia, D. Kozakov, and S. Vajda, "Performance and its limits in rigid body protein-protein docking," *Structure*, vol. 28, no. 9, pp. 1071–1081, 2020.

[39] D. Kozakov, R. Brenke, S. R. Comeau, and S. Vajda, "Piper: an fft-based protein docking program with pairwise potentials," *Proteins: Structure, Function, and Bioinformatics*, vol. 65, no. 2, pp. 392–406, 2006.

[40] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. a. Swaminathan, and M. Karplus, "Charmm: a program for macromolecular energy, minimization, and dynamics calculations," *Journal of computational chemistry*, vol. 4, no. 2, pp. 187–217, 1983.

[41] V. Feoktistov, *Differential Evolution: In Search of Solutions*. NY: Springer, 2006.

[42] D. Varela and J. Santos, "Crowding differential evolution for protein structure prediction," in *International Work-Conference on the Interplay Between Natural and Artificial Computation*, pp. 193–203, Springer, 2019.

[43] D. Varela and J. Santos, "Protein structure prediction in an atomic model with differential evolution integrated with the crowding niching method," *Natural Computing*, pp. 1–15, 2020.

[44] N. A. Marze, S. S. Roy Burman, W. Sheffler, and J. J. Gray, "Efficient flexible backbone protein–protein docking for challenging targets," *Bioinformatics*, vol. 34, no. 20, pp. 3461–3469, 2018.

[45] S. Chaudhury, S. Lyskov, and J. J. Gray, "Pyrosetta: a script-based interface for implementing molecular modeling algorithms using rosetta," *Bioinformatics*, vol. 26, no. 5, pp. 689–691, 2010.