GECONTEC: Revista Internacional de Gestión del Conocimiento y la Tecnología. ISSN 2255-5648
Bolufé-Röhler, A., Coto-Santiesteban, A., Soto, M.R. and Chen, S. Vol.2 (3). 2014

# Minimum Population Search, an Application to Molecular Docking

Antonio Bolufé-Röhler
bolufe@matcom.uh.cu
*Universidad de La Habana*
Alex Coto-Santiesteban
a.coto@lab.matcom.uh.cu
*Universidad de La Habana*
Marta Rosa Soto
mrosa@icimaf.cu
*ICIMAF*
Stephen Chen
sychen@yorku.ca
*York University*

## ABSTRACT

Computer modeling of protein-ligand interactions is one of the most important phases in a drug design process. Part of the process involves the optimization of highly multi-modal objective (scoring) functions. This research presents the Minimum Population Search heuristic as an alternative for solving these global unconstrained optimization problems. To determine the effectiveness of Minimum Population Search, a comparison with seven state-of-the-art search heuristics is performed. Being specifically designed for the optimization of large scale multi-modal problems, Minimum Population Search achieves excellent results on all of the tested complexes, especially when the amount of available function evaluations is strongly reduced. A first step is also made toward the design of hybrid algorithms based on the exploratory power of Minimum Population Search. Computational results show that hybridization leads to a further improvement in performance.

**KEY WORDS:** Minimum Population Search, Molecular Docking, Heuristic Algorithms, Optimization, Multi-modality.

## INTRODUCTION

Several population-based heuristics such as Particle Swarm Optimization (Bratton and Kennedy, 2007), Differential Evolution (Storn and Price, 1997) and Nelder-Mead (Lagarias et al., 1998) use line segments (e.g. difference vectors, attraction vectors, mid-point crossover, etc.) to generate new solutions. If the population size ($n$) of such algorithms is smaller than the dimensionality of the problem ($d$), then solutions generated through the mere combination of line segments will get trapped inside the $n$-1 dimensional hyperplane defined by the population members. The Nelder-Mead algorithm is a clear example: a simplex of $n = d+1$ solutionsis required to search a $d$-dimensional search space. To guarantee the effectiveness of the difference/attraction vectors, the recommended population size for PSO and DE is usually larger than the dimensionality of the problem (Mallipeddi and Suganthan, 2008). However, large populations may affect a metaheuristic's scalability and/or its ability to converge with a limited budget of function evaluations

(FEs). Conversely, if the population is too small, then the metaheuristic may be prone to premature convergence and may even be unable to cover the entire search space (i.e. when $n<d$ as is often the case for large $d$) (Chen et al., 2014).

Minimum Population Search (MPS) is a new metaheuristic explicitly designed to deal with this limitation (Bolufé-Röhler and Chen, 2014). The innovative idea of MPS is to provide strong exploration/diversification mechanisms which allow it to effectively cover the search space while using a (relatively) small population. In MPS the population size is set equal to the dimensionality of the problem ($n = d$). To generate new solutions, line segments are used to search within the $d$-1 dimensional hyperplane, and full coverage of the search space is then achieved by taking a subsequent step that is orthogonal to this hyperplane. To preserve the diversity of the (small) population and avoid premature convergence, the *thresheld convergence* technique is used. By establishing a minimum search step, thresheld convergence disallows new solutions which are too close to members of the current population, and this ensures a strong exploration during the early stages of the search. The minimum step (threshold) decays as the search progresses and convergence is thus "held" back until the last stages of the search process.

Previous work on the BBOB set of benchmark functions confirms the ability of MPS to perform effectively on multi-modal search spaces with a highly reduced budget of FEs (Bolufé-Röhler and Chen, 2013) (Bolufé-Röhler and Chen, 2014). This efficient use of the available computational effort/function evaluations may become useful in real-world applications where computing the objective function can be very time consuming. The molecular docking problem is one such real-world applicationwhere the objective function is highly multi-modal as well as costly to compute. Thus, we consider it to be a good real-world problem to test the performance of MPS.

In addition to comparing MPS against a wide range of algorithms on the molecular docking problem, this work also presents a first approach for hybridizing MPS. A distinctive feature of MPS is providing a strong exploration with a small population. It allows detecting promising regions of the search space with a highly reduced budget of function evaluations. If stopped before converging, the population of MPS may become a good starting point for heuristics with a stronger local search strategy. Simple hybrids based onheuristics such as Covariance Matrix Adaptive Evolutionary Strategy (CMA-ES) (Hansen et al., 2003), Nelder-Mead (NM)(Lagarias et al., 1998) and Pattern Search (PS) (Audet, and Dennis, 2003)are presented and analyzed. Computational results demonstrate the potential for MPS hybrids as the hybrid algorithms outperform both MPS and the (stand-alone) method used to find a local optimum.

The next section describes the molecular docking problem. Section III presents a background on the optimization of multimodal functions and thresheld convergence. In Section IV, MPS is introduced from its basic ideas to its current state of the art. Computational results for MPS and a broad range of other metaheuristics are presented in Section IV. The hybrids of MPS using the standard and adaptive threshold functions are presented in Section V and VI, respectively. Conclusions about the research are finally presented in Section VII.

## MOLECULAR DOCKING

Molecular docking is a computational method which aims to predict the three-dimensional structure of a protein-ligand complex. In the field of rational drug design, solution of the docking problem is of great value, as it allows prediction of binding conformations and affinities between drug molecules and target proteins (Brooijmans and Kuntz, 2003). The docking problem has been extensively addressed, both academically and commercially (DOCK, AutoDock, GOLD, ICM, and FlexX). However, docking is still a time-consuming task even with the most powerful computing resources currently available. For these reasons, the development of faster and more accurate docking methods remains an active field of research (Trott and Olson, 2009).

There are two key components to docking. First, there is the search strategy which is in charge of sampling the configuration space. Optimization algorithms are frequently used for this purpose. Second, there is the "scoring function" which allows the ranking and selection of the best solutions found by the search algorithm. Generally, these scoring functions are approximations of the free binding energy of the complex, and they allow finding the correct binding mode by assuming that the configuration with the lowest energy is the "right" one (i.e. the experimentally observable one). Ideally, a scoring function should be as simple as possible to compute, have few local minima, and its global optimum should correspond to the correct binding conformation (Kitchen et al., 2004).

The correct binding mode can usually be found by using methods like X-ray crystallography and NMR spectroscopy, but as docking simulations are much cheaper and faster, these methods complement each other in practice(Haile, 1992). Large sets of complexes can be sampled by using docking first, and then experimentally tested by using one of the aforementioned methods. The experimentally observed structure can be used to test the accuracy of a docking method through the root mean square deviation (RMSD) measure (Kroemer, 2007). A solution with a RMSD less than 2 Å is classified as docked and it is considered a very good result. A solution with a RMSD less than 3 Å is classified as partially docked (Magalhães et al., 2004).

Many approximations are made in order to speed up docking. One of the most common approaches is to assume that the receptor conformation does not change throughout the docking process. This approach highly reduces the dimensionality of the problem. Even so, the scoring function remains the most computationally expensive component in docking algorithms. Scoring functions are also highly multimodal and rugged, so they are difficult to optimize (Tavares et al., 2002). Therefore, previous work has used several heuristics such as Simulated Annealing (Goodsell and Olson, 1990), Iterated Local Search (Trott and Olson, 2009), Genetic Algorithms (Tavares et al., 2002), Differential Evolution (Thomsen, 2003), Particle Swarm Optimization (Namasivayam and Günther, 2007), among others.

***Table 1: Description of the complexes used in the experiments***

| PDB code | Protein receptor | Box dimensions (A) | Ligand torsions |
|---|---|---|---|
| 1adb | Alcohol dehydrogenase | 28×20×22 | 15 |
| 1bmm | Alpha-Thrombin | 17×19×22 | 10 |
| 1cjw | Serotonin N-acetyltransferase | 26×22×30 | 26 |
| 2z5u | Lysine-specific histone demethylase 1 | 28×32×24 | 20 |

Population-heuristics are generally reported to obtain better results than algorithms that consider only one conformation at a time (Halperin, 2002). Hybridization of global and local search algorithms is also a very frequent strategy used to improve docking accuracy (Thomsen, 2003).

For our experiments, we will be using the four complexes described in Table 1. These were examples selected because of their relative high number of rotatable bonds, since it is harder for current search algorithms to deal with these types of complexes. The scoring function used is the Autodock 4 scoring function. Autodock is a flexible software that is available free for academic usage, and it is frequently used in docking research (Morris et al., 2009).

## BACKGROUND

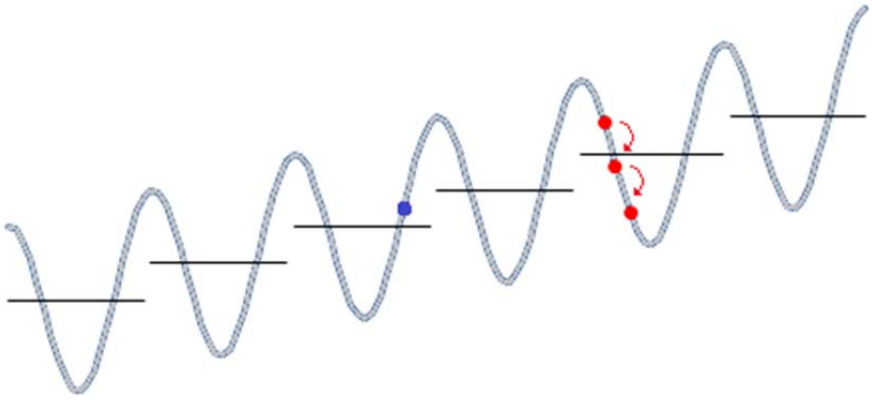
In multi-modal search spaces, following a gradient doesn't necessarily lead the search in a good direction. Thus, a general guideline for optimization in these search spaces is to perform an initial exploratory search to detect the most promising regions. Once these regions have been found, local search is required to converge to the corresponding (local) optima. A limitation of this approach is that usually a large amount of function evaluations are required to effectively perform both processes of global and local search. Increasing the efficient use of FEs is thus an important goal in multi-modal search spaces.

In evolutionary algorithms, the population size is a crucial parameter strongly related with the efficient use of FEs (Grefenstette, 1986). Larger populations, for instance, allow more exploration and reduce the risk of premature convergence and stagnation. However, evolving a large population usually requires larger budgets of FEs to achieve convergence. With smaller populations, efficiency in the use of FEs is increased and convergence can be achieved with fewer evaluations, but the risk of premature convergence and stagnation also rises. Many research works have been conducted to develop population-based algorithms with small populations, also called micro-algorithms.

The key concern when using a small population is how to increase/maintain diversity and exploration. Previous research has applied the idea of reducing the population size to a wide range of evolutionary algorithms such as Estimation Distribution Algorithms (EDA) (Hong et al., 2007), Genetic Algorithms (GA) (Reeves, 1993), Differential Evolution (Caraffini et al., 2013), Particle Swarm Optimization (Cabrera and Coello, 2010) and multi-swarm systems (Bolufé-Röhler and Chen, 2011). The increased efficiency achieved by these algorithms have proven to be useful in fields such as Interactive Evolutionary Computation (IEC) (Takagi, 2001), Multi-Objective Optimization (MOO) (Cabrera and Coello, 2010) and Large Scale Global Optimization (LSGO) (Brest et al., 2008). In each case, explicit techniques to preserve the diversity of the population were developed.

Crowding and niching are among the most popular diversification techniques. These methods allow allocating and maintaining multiple optimal/suboptimal solutions in a population (Thomsen, 2004). Inspiration for niching stems from nature, where diverse species coexist through adaptation to different niches. To maintain diversity, explicit mechanisms are employed to split the search efforts across different parts of the search space. To achieve this, niching frequently involves the use of minimally-interacting subpopulations. Crowding, on the other hand, maintains diversity by comparing each new solution against a subset of the population and replacing the one most similar to it (Brits et al., 2002).

***Figure 1: A solution found through local search (red) may look more promising than a random solutionfrom a better attraction basin (blue).***



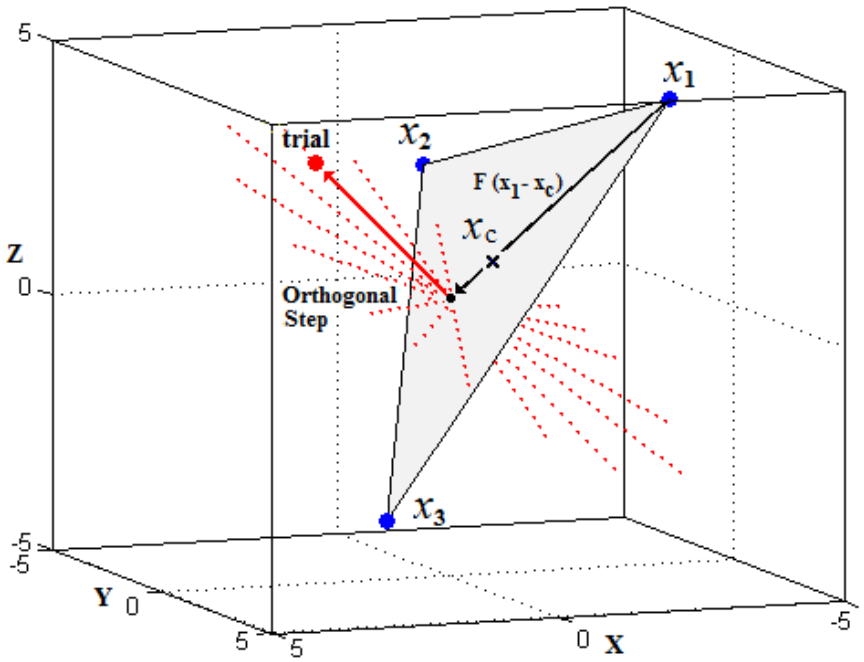
Although conceived to maintain a diversified population, these methods do not address the effects of having global and local search occurring concurrently during the search process. In many heuristics, large (global) and small (local) search steps are made at the same time. As a result, the process of detecting the best regions may be compromised. Better solutions are likely to be found in attraction basins (regions in which greedy descent will lead to the same local optimum) where several small/local search steps have been performed. This attraction basin may then look more promising than a better attraction basin where less local search has been done. Figure 1 illustrates how a solution found through excessive local search (red) may have a better fitness, despite being located in a worse attraction basinthan a randomly picked solution (blue) from a better attraction basin.

The thresheld convergence (TC) technique has been developed to address this issue (Chen and Montgomery, 2011). The key idea is to avoid small steps (local search) during the early stages of the search process. This allows the search method to sample, without bias, different attraction basins. Once enough exploration has been performed, a transition into local search occurs in order to find the local optima of the best detected attraction basins. Thresheld convergence achieves this by enforcing a minimum size on the search step. The minimum step (threshold) decays as the search progresses and convergence is thus "held" back until the last stages of the search process. By controlling the decay rate of the threshold function, it is possible to effectively determine the amount of exploration and exploitation performed by the algorithm(Bolufé-Röhler et al. 2013).

## MINIMUM POPULATION SEARCH

Minimum Population Search is a recently developed metaheuristic specifically designed for optimizing multi-modal functions. The key idea is to guarantee a full search into all dimensions using the minimum required population size. Similarly to other population-based heuristics, MPS uses line segments as its main search mechanism. However, if the population size becomes smaller than the dimensionality of the problem, line segments will be restricted to the $n$-1dimensionalhyperplane formed by the $n$ population members. To overcome this limitation the population size is set equal to the dimensionality of the problem ($n = d$), and a subsequent step orthogonal to the $d$-1 dimensional hyperplanethen guarantees searching into all dimensions. (Bolufé-Röhler and Chen, 2014).

***Figure 2: Generation of a new solution (trial) in MPS.***




Every generation a new solution ($trial_i$) is created from each population member ($x_i$). First, from each parent solution $x_i$ a step inside the *d-1*hyperplane (formed by the *n* population members) is performed. Then, an orthogonal step is made to search into the missing dimension. The "in-plane" step is made using the difference vector between the parent solution ($x_i$) and the centroid of the current population ($x_c$). The orthogonal step is made taking a random vector orthogonal (*orth*) to the parent-centroid difference vector (Figure 2). This two-step process for generating the new trial solutions ($trial_i$) is represented in (1). The direction and size of the difference and the orthogonal vectors are determined by the scaling factor $F_i$ and $O_{step\_i}$, respectively.

$$trial_i = x_i + F_i * (x_i - x_c) + O_{step\_i} * orth \tag{1}$$

To promote diversification,thresheld convergence forces new solutions to be a minimum (*min_step*) threshold distance away from their parent solutions. To avoid new solutions from being sampled too far away from the best found regions, MPS also enforces a maximum search threshold (*max_step = 2 * min_step*). To guarantee that the difference vector step doesn't exceed the maximum allowed threshold distance, the $F_i$ factor is drawn with a uniform distribution from [*-max_step, max_step*]. To ensure that the new solution ($trial_i$) falls in the correct (*min_step, max_step*) threshold interval,the $O_{step\_i}$ factor is selected with a uniform distribution from [$min\_orth_i$, $max\_orth_i$]. The $min\_orth_i$ and $max\_orth_i$ values are calculated using (2) and (3), respectively. The difference vector ($x_i$-$x_c$) and the orthogonal vector (*orth*) are normalized before scaling. Once the new solutions are created, clamping is performed if necessary, and the best *n* solutions among the parents and offspring survive into the next generation.

$$min\_orth_i = \sqrt{\max(\min\_step_i^2 - F_i^2, 0)} \tag{2}$$

$$max\_orth_i = \sqrt{\max(max\_step_i^2 - F_i^2, 0)} \tag{3}$$

***Figure 3: Minimum Population Search Algorithm***

```
MPS (α,γ, maxFEs)
X ← InitialPopulation()                                   // Equation (5)
while FEs <maxFEs
        min_step ← UpdateThreshold(α,γ)                   // Equation (4)
        max_step ← 2 * min_step
        x_c ← CalculateCentroid()
        for i = 1 : popsize
                F_i ← UniformRandom(-max_step, max_step)
                orth_i ← OrthogonalVector(x_i- x_c)       // normalized vector
                orth_step ← UniformRandom(min_orth, max_orth)   // Equations (2) and (3)
                trial_i ← x_i + F_i*(x_i - x_c) + orth_step*orth_i   // clamping if necessary
                Offpsring.Add(trial_i)
        endfor
        X ← BestSolutions(X, trial)
endwhile
```

The *min_step* values are updated by a rule similar to that used in previous attempts to control convergence for PSO (Chen and Montgomery, 2011) and DE (Montgomery and Chen, 2012) in which an initial threshold is selected that then decays over the course of the search process, see (4). Equation (4) shows how *min_step* is calculated:α represents a fraction of the main space diagonal, *FEs* is the total available amount of function evaluations, *k* is the number of evaluations used so far, and $\gamma$ is the parameter that controls the decay rate of the threshold. The current implementation uses α= 0.3 and $\gamma$ = 3, as suggested in (Bolufé-Röhler and Chen, 2013).

$$min\_step_i = \alpha * diagonal * ([FEs - k] / FEs)^{\gamma} \quad (4)$$

To ensure good spacing in the initial population, the initial points are selected to be on the diagonal of the search space. Assuming that the search space is bounded by the same lower and upper bound in each dimension, each population member is initialized using (6): $s_k$ is the *k-th* population member, $rs_i$ are random numbers which can be -1 or 1, and *bound* is the lower and upper bound in each dimension. This initialization method leads to a better distribution of the initial solutions in the search space than did uniform random solutions. A detailed pseudo-code is presented in Figure 3.

$$s_k = (rs_1 * bound / 2,\ rs_2 * bound / 2,\ ...,\ rs_n * bound / 2) \quad (5)$$

The molecular docking problem presents different (and asymmetric) bounds for each variable. Several components of MPS, such as the initialization method and the thresheld convergence minimum and maximum step, were designed having in mind equal bounds for variables, centered on the search space origin (as in most benchmark functions). Thus, to guarantee the correct functioning of MPS, the algorithm is executed with each variable bounded on [-1, 1]. Before evaluating the objective function, each solution is scaled back to the molecular docking search space using (6). In (6), $var_k$ is the value of the $k^{th}$ variable on the [-1, 1] search space, $range_k$ and $mid_k$ are respectively the range and middle point for the bounds of variable *k* in the molecular docking search space, and $md\_var_k$ is the value for variable *k* after scaling back to the molecular docking problem.

$$md\_\mathrm{var}_k = mid_k + range_k * \mathrm{var}_k \quad (6)$$

## COMPUTATIONAL RESULTS

A set of experiments has been designed to test the effectiveness of MPS on the molecular docking problem. The experiments include comparisons to other population-based and local search metaheuristics such as Genetic Algorithms, Differential Evolution, Particle Swarm Optimization and the Univariate Marginal Distribution Algorithm (UMDA), Pattern Search (PS), the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) and the simplex-based search method Nelder-Mead (NM).

The UMDA algorithm is a standard implementation using Gaussian density functions and truncation selection (Larrañaga and Lozano, 2011), the population size is $n = 200$ and the selection coefficient is $\epsilon=0.2$(Bolufé-Röhler and Chen, 2012). PSO is a standard version with ring topology (Bratton and Kennedy, 2007), with zero initial velocities (Engelbrecht, 2012) and "Reflect-Z" for particles that exceed the boundaries of the search space (Helwig et al., 2013). The DE method is the highly common and frequently effective variant labeled DE/rand/1/bin,from the implementation provided in (DE code, 2014). As recommended in (Bratton and Kennedy, 2007) and (Bolufé-Röhler and Chen, 2013), a population of $n = 50$ was used for DE and PSO. The CMA-ES code is the MATLAB version 3.61.beta available in (CMA-ES code, 2014), the default set of parameters was used. The Nelder-Mead and Pattern Search algorithms use the implementation provided by MATLAB with the default set of parameters.The Genetic Algorithm implementation is also from

***Table 2: Performance of different heuristics on the molecular docking problem***

| | | MPS | PSO | UMDA | GA | DE | CMAES | PS | NM |
|---|---|---|---|---|---|---|---|---|---|
| **1adb** | **Median** | -8.1e+0 | -8.1e+0 | **-8.9e+0** | -8.5e+0 | 2.4e+1 | **-8.9e+0** | -4.0e+0 | 1.3e+4 |
| | **Mean** | -7.3e+0 | -8.4e+0 | **-8.7e+0** | 3.2e+1 | 2.6e+1 | -5.6e+0 | 2.2e+3 | 6.6e+4 |
| | **Std.Dev.** | 6.7e+0 | 2.2e+0 | 6.2e+0 | 1.5e+2 | 2.2e+1 | 1.7e+1 | 8.6e+3 | 1.2e+5 |
| | **RMSD** | 2.3e+0 | 1.1e+0 | 2.1e+0 | **9.1e-1** | 6.5e+0 | 7.1e+0 | 2.8e+1 | 4.9e+1 |
| **1bmm** | **Median** | **-9.9e+0** | -9.4e+0 | -8.9e+0 | -7.9e+0 | -8.5e+0 | -7.6e+0 | -4.8e+0 | 1.9e+1 |
| | **Mean** | **-1.0e+1** | -9.6e+0 | -8.9e+0 | -7.7e+0 | -8.0e+0 | -5.4e+0 | 7.7e+1 | 1.0e+4 |
| | **Std.Dev.** | 1.5e+0 | 1.4e+0 | 1.4e+0 | 2.2e+0 | 1.9e+0 | 5.3e+0 | 5.5e+2 | 3.5e+4 |
| | **RMSD** | **3.2e+0** | 4.5e+0 | 6.9e+0 | 7.3e+0 | 4.4e+0 | 8.1e+0 | 1.3e+1 | 7.4e+0 |
| **1cjw** | **Median** | **-9.6e+0** | -5.9e+0 | -9.2e+0 | -4.7e+0 | 8.9e+1 | -9.3e+0 | 2.5e+1 | 1.2e+4 |
| | **Mean** | **-7.4e+0** | -5.2e+0 | 2.2e+1 | 3.7e+0 | 1.3e+2 | -1.5e-1 | 4.6e+3 | 8.0e+4 |
| | **Std.Dev.** | 1.3e+1 | 4.0e+0 | 1.8e+2 | 2.2e+1 | 1.2e+2 | 2.1e+1 | 1.8e+4 | 1.4e+5 |
| | **RMSD** | 5.9e+0 | **5.4e+0** | 8.9e+0 | 6.1e+0 | 6.2e+0 | 8.1e+0 | 7.3e+0 | 1.1e+1 |
| **2z5u** | **Median** | **-3.5e+0** | 2.0e+2 | 1.6e+1 | 2.5e+2 | 5.7e+3 | 3.7e+1 | 4.3e+3 | 1.4e+5 |
| | **Mean** | **3.3e+1** | 3.9e+2 | 1.1e+2 | 1.6e+3 | 9.7e+3 | 2.8e+2 | 3.8e+4 | 2.0e+5 |
| | **Std.Dev.** | 1.1e+2 | 5.1e+2 | 3.5e+1 | 5.0e+3 | 1.2e+4 | 5.9e+2 | 7.8e+4 | 1.9e+5 |
| | **RMSD** | **7.8e-1** | 2.6e+0 | 9.1e-1 | 8.5e-1 | 3.9e+0 | 5.7e+0 | 1.8e+1 | 9.7e+0 |

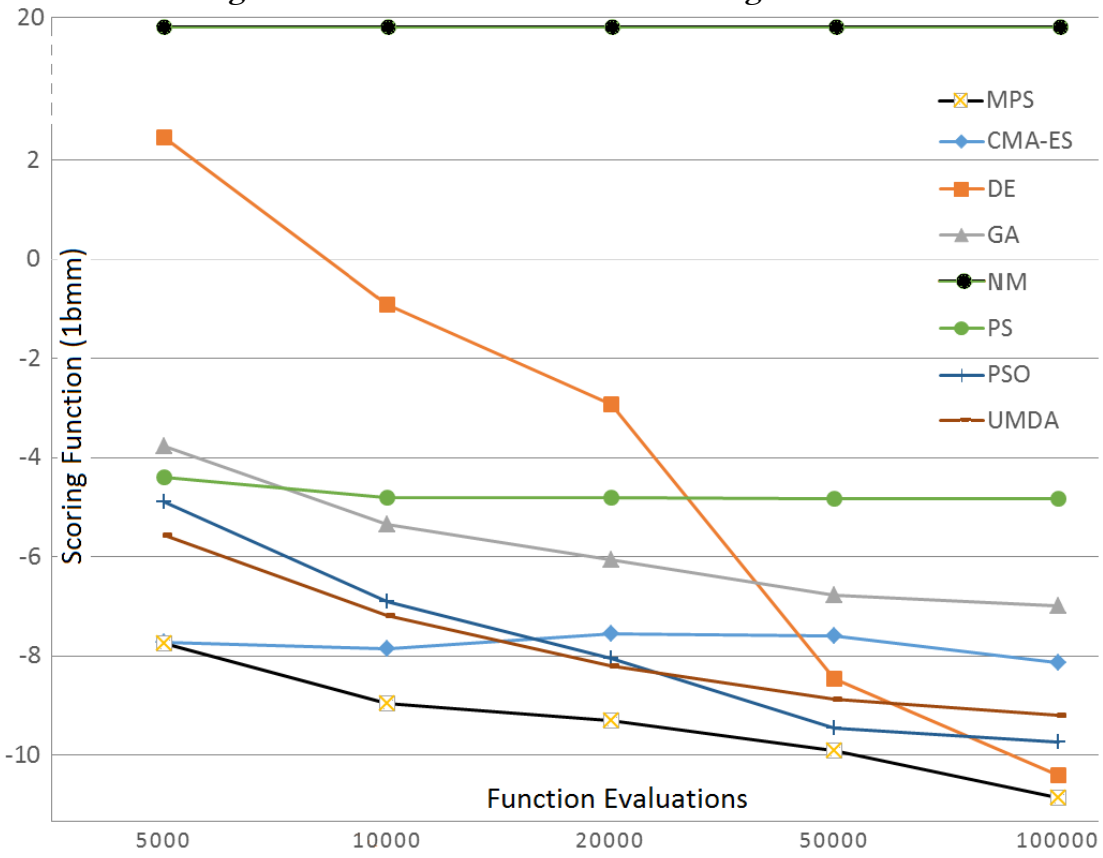
MATLAB using heuristic crossover, adaptive mutation, and a population size of 100 individuals. Minimum Population Search uses the parameters recommended in (Bolufé-Röhler and Chen, 2013), i.e. $a=0.3$ and $\gamma=3$ for the threshold function.

The amount of function evaluations is one of the most frequently used stop conditions, but on previous work this range is from 10.000 to 2.500.000. Taking into account that function evaluations in molecular docking are time consuming, results in Table 2 are presented with a maximum of 50.000 function evaluations, as in (Hou et al., 1999).Results in Table 2 include the mean, standard deviation and median over a 100 trials on every complex. The mean value for the root mean square deviation is also presented.

As it can be noticed MPS provides the best mean and median on 3 of the 4 complexes, providing a very solid performance among all the tested problems. On the 2z5u complex, MPS achieves an advantage of at least one order of magnitude against all of the other algorithms. The square root mean deviation achieved by MPS is the best one in 2 of the 4 complexes, and a successful docking (RMSD≤2.0Å) is achieved in 3 of the complexes. It is worth noticing the large difference between the reported mean and median values for all the tested methods. Due to the high multimodality, ruggedness, and deceptiveness of the scoring function, occasional executions of the algorithms get trapped at low quality local optima strongly affecting the overall mean performance. Thus, we consider the median as a more accurate measurement of the algorithms' performance.

Molecular calculations are in general very time consuming. Thus, achieving meaningful results with highly restricted budgets of FEs becomes important. Figure 4 presents the results of all the algorithms with varying budgets of allowed function evaluations for the

***Figure 4: Performance with variable budgets of FEs.***



1bmm complex. The plotted values are the median resultfrom 100 trials.As expected, the performance of most heuristics clearly drops as the budget of available FEs is reduced. However, CMA-ES, NM, and PS show a stable performance as the amount of evaluations is reduced. In the case of Pattern Search, this distinctive behavior is characteristic of single-point local search methods. Local search methods are usually greedier on their exploration of the search space allowing a faster convergence (with less FEs) to local optima. Similarly, the CMA-ES and Nelder-Mead algorithms are also known for their intense local search strategy and the ability to converge with a reduced budget of evaluations (Molina et al., 2010) (Liu and Yang, 2012). This characteristic suggests that PS, NM, and CMA-ES could be effectively used as local exploitation methods in a hybrid algorithm (Payne and Eppstein, 2005) (Gao et al., 2011).

## HYBRIDIZING MINIMUM POPULATION SEARCH

A distinctive feature of MPS is providing a strong exploration with a small population. It allows detecting promising regions of the search space at early generations with a strongly reduced budget of function evaluations. Although MPS starts performing local search as the threshold size decreases, its search mechanism is highly exploratory (e.g. orthogonal step). Thus, if stopped before converging, the population of MPS may become a good starting point for heuristics with a stronger local search strategy.

A simple hybrid algorithm was designed to confirm this hypothesis. First, MPS is executed and stopped once it reaches half of the FEs budget. Stopping before consuming all of the function evaluations keeps the threshold from becoming too small and MPS from performing local search. The rest of the FEs are then used to perform a greedier/local search. Starting from the best found solution the CMA-ES, PS, and NM methods are executed. If further FEs are available after the method converges, the second best solution is used as the starting point, and so on until the whole budget of function evaluations is consumed. Although CMA-ES and NM are population-based algorithms, the implementations provided in (CMA-ES code, 2014) and MATLAB, respectively, allow the algorithm to startfrom a singleinitial solution.

Three hybrids of MPS using CMA-ES, PS, and NM were tested. Table 3 shows the median results of the three hybrids for 100 trials with a budget of 50.000 FEs. The relative performances (%-diff = (a-b)/max(a,b)) achieved by each hybrid versus MPS and the hybridizing heuristic are also shown. These values indicate by what amount (percent) the given hybrid (b) outperforms each of the hybrid's components (a). The compared values

***Table 3: Relative performance of the MPS hybrids***

| | **MPS-CMAES** | | | **MPS-PS** | | | **MPS-NM** | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Median** | **%-diff (MPS)** | **%-diff (CMAES)** | **Median** | **%-diff (MPS)** | **%-diff (PS)** | **Median** | **%-diff (MPS)** | **%-diff (NM)** |
| **1adb** | **-9.3e+0** | 9.4% | 2.8% | -8.6e+0 | 4.4% | 27.2% | -8.6e+0 | 4.3% | 99.9% |
| **1bmm** | -9.9e+0 | 0.4% | 27.8% | -9.9e+0 | 0.6% | 45.8% | **-1.1e+1** | 3.0% | 83.4% |
| **1cjw** | -9.8e+0 | 1.5% | 3.8% | -9.8e+0 | 1.9% | 73.9% | **-1.1e+1** | 11.4% | 99.9% |
| **2z5u** | -2.7e+0 | -2.8% | 58.5% | -2.6e+0 | -3.0% | 99.3% | **-8.7e+0** | 18.9% | 99.9% |
| **Mean %diff.** | | **2.1%** | **23.2%** | | **1.0%** | **61.6%** | | **9.4%** | **95.8%** |

(a) and (b) are the errors (difference) between the best known solution and the achieved median value.

Results in Table 3 confirm that MPS may benefit from a greedier search during the last stages of the optimization process. In general, the mean relative improvement over the local search methods is very large, rising up to an impressive 95.8% when compared to Nelder-Mead. This result and the large standard deviation on Table 2 suggest that Nelder-Mead is the local search method with the strongest dependence on the initial solutions. However, when provided when a good initial guess, Nelder-Mead can be an effective and efficient technique for reaching nearby (local) optima. The hybrid between MPS and Nelder-Mead provides the best performance, achieving the best results on 3 of the 4 complexes. The MPS-NM hybrid also achieves the largest relative improvement when compared to MPS with an almost 10% average improvement.

## ADAPTIVE THRESHOLD FUNCTION

To help local search find different local optima, the final solutions of MPS (i.e. the initial solutions for CMA-ES, PS, and NM) should be located in different attraction basins. If the distance among the different attraction basins could be known *a priori*, then the threshold could be held longer at this "ideal search scale". Once the threshold is at this search scale MPS could be stopped before solutions start falling inside the same attraction basins. However, the size of the attraction basins is not the same for all problems, and it thus becomes difficult to predict when to stop MPS.

An alternative approach is to use an adaptive threshold function. By adaptively adjusting γ, it is possible to control the convergence speed and stabilize the threshold at the "ideal search scale". A simple strategy can be used to adjust γ. If an improvement is achieved, i.e. at least one offspring survives to the next generation, it suggests that exploration is paying off and convergence should be delayed – subsequently γ is decreased. Otherwise, if no improvements are achieved, it implies that the current search scale has been sufficiently explored so more local search may now be necessary – subsequently γ is increased (Bolufé-Röhler and Chen, 2014).

***Figure 5: Standard and adaptive threshold functions***

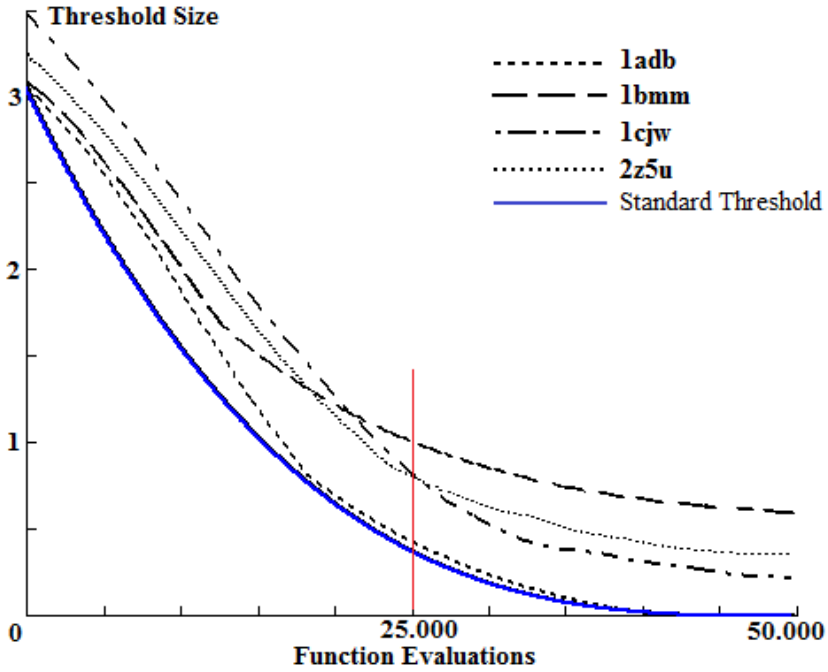


Figure 5 shows the adaptive threshold decay rate (minimum_step) for each of the four tested complexes. The standard (non-adaptive) threshold function is also shown. The (initial) threshold parameters are $\alpha = 0.1$ and $\gamma = 3$ and the step size to increase/decrease $\gamma$ is 0.005. The threshold function is plotted for a single trial of 50.000 function evaluations. As it can be seen, on most complexes, the threshold value of the adaptive function decreases until it stabilizes at a potentially "ideal search scale". This "ideal" scale is appropriately different for each complex. When 50% of the FEs budget is consumed (vertical red line), MPS is stopped and the heuristics with a stronger local search ability take over. It can be noticed that at this point the threshold value is above the final search scale (at 50,000 FEs), and this should ensure that the MPS solutions are still located in different attraction basins.

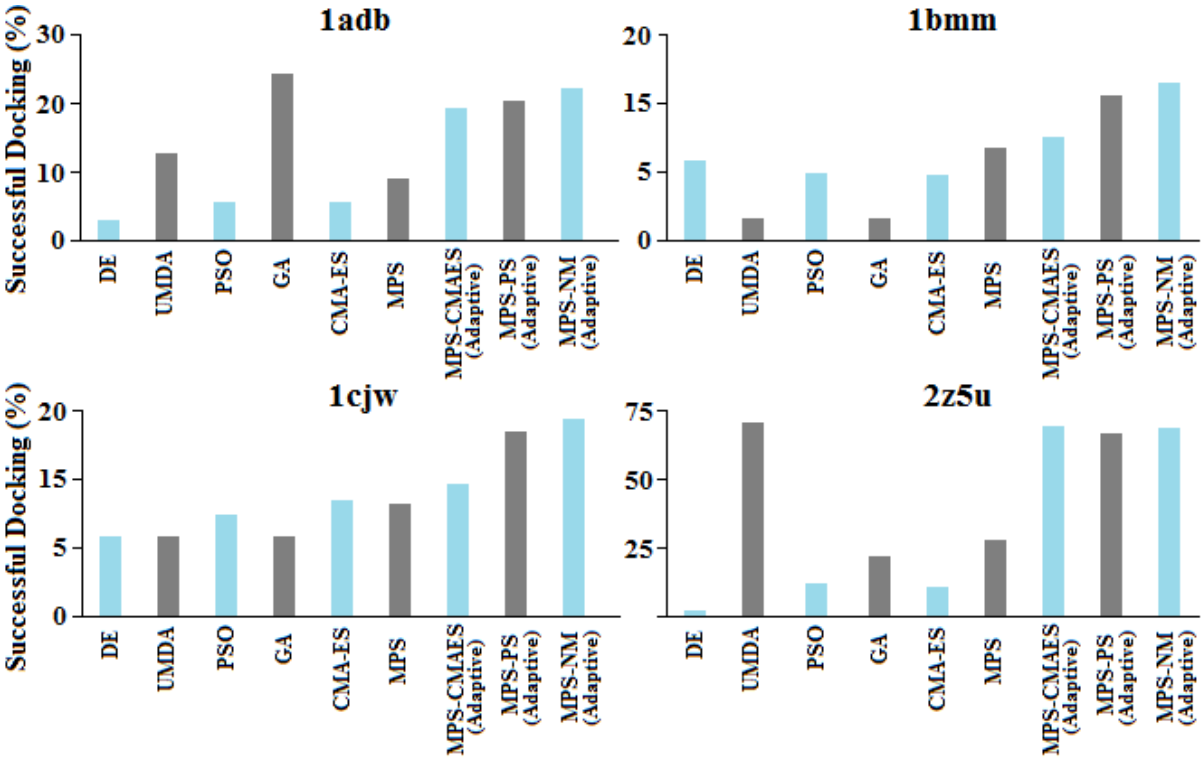
Table 4 shows the results of the MPS hybrids when using the adaptive threshold function. If compared to hybrids using the standard threshold function (Table 3), it can be noticed that performance increases for all hybrids on all complexes. The three hybrids show an overall improvement of approximately 25% when compared to MPS. The maximum improvement is achieved on the 2z5u complex. On this complex MPS shows the best performance compared to other heuristics (Table 2), but the non-adaptive MPS_CMAES and MPS_PS hybrids also have a decrease in performance compared to MPS (Table 3). These results suggest that MPS can be further improved through hybridization with greedier heuristics, but the threshold size has to be kept large enough to avoid solutions from falling inside the same attraction basins (which would eliminate the benefit of multiple local optimizations).

In practice, the effectiveness of molecular docking algorithms is frequently measured by their ability to achieve a successful docking (i.e. an RMSD≤2.0Å). Figure 6 shows the amount of successful docking (percent) achieved by each heuristic in every complex. Results for Nelder-Mead and Pattern Search are not presented because for most complexes they don't achieve a successful docking. In Figure 6, it can be noticed that the performance of the simple heuristics strongly varies depending on the tested complex. Nevertheless, MPS shows a stable performance being always among the best three simple heuristics. Furthermore, when hybridized the performance strongly improves on all the tested complexes. For completeness, the mean, standard deviation, and average RMSD for the adaptive hybrids are presented in Table 5.

***Table 4: Relative performance of the adaptive MPS hybrids***

| | **MPS-CMAES (adaptive)** | | | **MPS-PS (adaptive)** | | | **MPS-NM (adaptive)** | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Median** | **%-diff (MPS)** | **%-diff (CMAES)** | **Median** | **%-diff (MPS)** | **%-diff (PS)** | **Median** | **%-diff (MPS)** | **%-diff (NM)** |
| **1adb** | **-1.1e+01** | 19.9% | 14.0% | -1.0e+01 | 17.1% | 36.9% | -1.0e+01 | 17.5% | 99.9% |
| **1bmm** | **-1.1e+01** | 12.9% | 36.9% | -1.1e+01 | 10.7% | 51.4% | -1.1e+01 | 11.4% | 84.8% |
| **1cjw** | -1.2e+01 | 16.5% | 18.4% | -1.3e+01 | 27.1% | 80.6% | **-1.3e+01** | 27.6% | 99.9% |
| **2z5u** | -1.5e+01 | 42.5% | 76.8% | -1.6e+01 | 44.5% | 99.6% | **-1.7e+01** | 50.3% | 99.9% |
| **Mean %diff.** | | **23.0%** | **36.5%** | | **24.9%** | **67.1%** | | **26.7%** | **96.2%** |

***Figure 6: Successful Docking***




## CONCLUSSIONS

Minimum Population Search has been shown to be an effective heuristic for solving molecular docking problems, and it can outperform many of the algorithms commonly used for this task. The ability to provide full coverage of the search space with the smallest population allows MPS to perform more generations and to provide good results with reduced budgets of function evaluations. The balance between global and local search provided by thresheld convergence disallows premature convergence, and this helps MPS to deal with the deceptiveness, ruggedness, and high multi-modality of the docking problem.

The optimization of multi-modal problems involves two distinct tasks: identifying promising attraction basins and finding the local optima in these basins. To effectively perform each of these tasks, heuristics can benefit from different search strategies. To identify promising attraction basins, a more global/exploratory approach is useful to avoid bias and premature convergence towards low-quality attraction basins. On the other hand, finding a local optimum in a given attraction basin benefits from a strong local/exploitive search strategy to promote a fast convergence to the optimum. The hybridization of MPS tackles this issue by assigning a different heuristics to each task. The search for promising attraction basins is performed with MPS while the convergence to the corresponding local optima is done through greedier heuristics with stronger local search capabilities.

The promise of an attraction basin is often estimated by the fitness of a single sample solution, so an attraction basin represented by a random sample solution can appear to be less promising than an attraction basin represented by its local optimum. The use of thresheld convergence as a central component in MPS aims to avoid these biased comparisons by disallowing local search while global search is still in progress. Ideally, the

***Table 5: Performance of the adaptive hybrids on the molecular docking problem***

| | MPS-CMAES | | | MPS-PS | | | MPS-NM | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Mean** | **Std.Dev.** | **RMSD** | **Mean** | **Std.Dev.** | **RMSD** | **Mean** | **Std.Dev.** | **RMSD** |
| **1adb** | **-1.1e+01** | 7.5e+0 | 1.6e+0 | **-1.1e+01** | 3.0e+0 | 2.0e+0 | **-1.1e+01** | 2.1e+0 | 1.4e+0 |
| **1bmm** | **-1.1e+01** | 1.9e+0 | 3.8e+0 | **-1.1e+01** | 1.6e+0 | 1.6e+0 | **-1.1e+01** | 1.5e+0 | 1.5e+0 |
| **1cjw** | -1.0e+01 | 1.8e+1 | 3.0e+0 | **-1.3e+01** | 1.3e+1 | 2.8e+0 | **-1.3e+01** | 3.0e+0 | 2.9e+0 |
| **2z5u** | **-9.5e-01** | 2.4e+2 | 1.1e+0 | 1.2e+01 | 1.5e+2 | 4.7e-1 | 7.7e+00 | 9.9e+1 | 4.7e-1 |

threshold size (i.e. minimum and maximum step size) is correlated to the size of the attraction basins in the search space. The minimum step size should avoid sampling solutions from the same attraction basins, while the maximum step size should restrict the sampling of (low-quality) attraction basins too far away from the current search region. However, predicting this "ideal search scale" is very difficult without an *a priori* knowledge of the objective function's landscape.

The use of an adaptive threshold function attempts to dynamically identify the scale of the search space. By holding the minimum step above this search scale the chances of oversampling an attraction basin are strongly reduced. This promotes a more exploratory search and increases the chances of detecting the most promising attraction basins. If MPS is stopped while the minimum step is above this scale then different solutions will most likely be located in different attraction basins. Using heuristics with greedier search capabilities and faster convergence rates allow function evaluations to be saved – the local optima of more attraction basins can thus be found within the given budget of FEs. This combination of an adaptive threshold function and hybridization has led to an average improvement of approximately 25% over standard MPS on the current problem set.

Future work will focus on extending these results towards large scale global optimization (LSGO). In LSGO the key challenge is the exponential increase of the search space volume versus the linear increase of function evaluations. Known as "the curse of dimensionality" (Bellman, 1957), these mismatched growth rates between search space volume and available FEs can cause adequate search space coverage in low dimensional search spaces to become exceptionally sparse coverage in high dimensional search spaces. Algorithms for solving high-dimensional optimization problems thus need to be efficient in the use of FEs and wisely exploit response surfaces (Chu et al., 2011). Through the hybridization of MPS with greedier (local search) heuristics, these conditions are met on both tasks of the multi-modal optimization process.

## REFERENCES

Audet, Charles and J. E., Dennis (2003). Analysis of Generalized Pattern Searches. *SIAM Journal on Optimization*, Vol. 13(3): 889–903.

Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press.

Bolufé-Röhler, A. and Chen, S. (2011). An analysis of sub-swarms in multi-swarm systems. AUS-AI, p.271–280.

Bolufé-Röhler, A. and Chen, S. (2012). Multi-swarm hybrid for multi-modal optimization. IEEE CEC, p. 1759–1766.

Bolufé-Röhler, A. and Chen, S. (2013). Minimum Population Search – Lessons from building a heuristic technique with two population members. IEEE CEC, p. 2061–2068.

Bolufé-Röhler, A., Estévez-Velarde, S., Piad-Morffis, A., Chen, S. and Montgomery, J. (2013). Differential evolution with thresheld convergence. IEEE CEC, p. 40–47.

Bolufé-Röhler, A., and Chen, S. (2014). Extending Minimum Population Search towards Large Scale Global Optimization. IEEE CEC. In Press.

Bratton, D. and Kennedy, J. (2007). Defining a standard for particle swarm optimization. IEEE SIS, p. 120–127.

Brest, J., Zamuda, A., Bošković, B., Maučec, M. S. and Žumer, V. (2008). High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction. IEEE CEC, p. 2032–2039.

Brits, R., Engelbrecht, A. P. and Van den Bergh, F. (2002). A niching particle swarm optimizer. SEAL, p. 692–696.

Brooijmans N. and Kuntz, I. (2003). Molecular Recognition and Docking Algorithms. Annual Review of Biophysics and Biomolecular Structure, p. 335–373.

Cabrera, F. J. and Coello, C. C. (2010). Micro-MOPSO: A multi-objective particle swarm optimizer that uses a very small population size. Multi-Objective Swarm Intelligent Systems, p. 83–104.

Caraffini, F., Neri, F. and Poikolainen, I. (2013). Micro-differential evolution with extra moves along the axes. IEEE Symposium on Differential Evolution, p. 46–53.

Chen, S. and Montgomery, J. (2011). A simple strategy to maintain diversity and reduce crowding in particle swarm optimization. Australasian AI, p. 281–290.

Chen, S., Montgomery, J. and Bolufé-Röhler, A. (2014). Some Measurements on the Effects of the Curse of Dimensionality. ACM GECCO, in Press.

Chu, W., Gao, X., and Sorooshian, S. (2011). A new evolutionary search strategy for global optimization of high-dimensional problems. *Information Sciences*, Vol. 181, Nov, p. 4909–4927.

Engelbrecht, A. (2012). Particle swarm optimization: velocity initialization. IEEE CEC, p. 70–77.

Gao, Z., Xiao, T. and Fan, W. (2011). Hybrid differential evolution and Nelder–Mead algorithm with re-optimization. *Soft Computing*, Vol. 15 (3): 581-594.

Goodsell, D. and Olson, A. (1990). Automated Docking of Substrates to Proteins by Simulated Annealing. Proteins: Structure, Function and Genetics, p. 195–202.

Grefenstette, J.J. (1986). Optimization of control parameters for genetic algorithms. IEEE SMC: 16(1): 122–128.

Haile J. M. (1992). Molecular Dynamics Simulations: Elementary Methods. Wiley, New York, p. 11–18.

Halperin I., Ma, B., Wolfson, H. and Nussinov, R. (2002). Principles of Docking: An Overview of Search Algorithms and a Guide to Scoring Functions. *Proteins* Vol. 47(4): 409-443.

Hansen, N., Muller, S. D. and Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evolutionary Computation, Vol. 11(1): 1–18.

Helwig, S., Branke, J. and Mostaghim, S. (2013). Experimental Analysis of Bound Handling Techniques in Particle Swarm Optimization. IEEE Trans. Evolutionary Computation, Vol. 17: 259–271.

Hong, Y., Kwong, S., Ren, Q. and Wang, X. (2007). Over-Selection: An attempt to boost EDA under small population size. IEEE CEC, p. 1075–1082.

Hou , T.J., Wang, J.M. and Xu, X. J. (1999). A Comparison of Three Heuristic Algorithms for Molecular Docking. Chinese Chemical Letters, Vol. 10(79): 615–618.

Kitchen, D.B., Decornez, H., Furr, J.R. and Bajorath, J. (2004). Docking and scoring in virtual screening for drug discovery: methods and applications. *Nat. Rev. Drug Discov.* Vol. 3(11): 935–949.

Kroemer R. (2007). Structure-Based Drug Design: Docking and Scoring. *Current Protein and Peptide Science*, Vol. 8: 312–328.

Lagarias, J. C., Reeds, J. A., Wright, M. H. and Wright, P. E. (1998). Convergence properties of the Nelder-Mead simplex method in low dimensions. *SIAM J. Optimization,* Vol. 9: 112–147.

Larrañaga P. and Lozano, J. A. (2011). Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, p. 181–193.

Liu, A. and Yang, M. (2012). A New Hybrid Nelder-Mead Particle Swarm Optimization for Coordination Optimization of Directional Overcurrent Relays. Mathematical Problems in Engineering.

Magalhães, C. S., Barbosa, J.C. and Laurent, E. (2004). A genetic algorithm for the ligand-protein docking problem.Genetics and Molecular Biology, Vol. 27 (4): 605–610.

Mallipeddi, R. and Suganthan, P. N. (2008). Empirical study on the effect of population size on differential evolution algorithm. IEEE CEC, p. 3663–3670.

Molina, D., Lozano, M., García-Martínez, C. and Herrera, F. (2010). Memetic Algorithms for Continuous Optimisation Based on Local Search Chains. Evolutionary Computation, *MIT Press*, Vol. 18 (1): 27–63.

Montgomery, J. and Chen, S. (2012). A simple strategy for maintaining diversity and reducing crowding in differential evolution. IEEE CEC, p. 2692–2699.

Morris, G. M., Huey, R., Lindstrom, W., Sanner, M. F., Belew, R. K., Goodsell, D. S. and Olson, A. J. (2009). Autodock4 and AutoDockTools4: automated docking with selective receptor flexiblity. *Journal Comput. Chem.*, Vol. 30: 2785–2791.

Namasivayam, V. and Günther R. (2007). PSO@AUTODOCK: A fast flexible molecular docking program based on swarm intelligence. *Chem. Biol. Drug Des.*, Vol. 70: 475–484.

Payne, J. L. and Eppstein, M. J. (2005). A hybrid genetic algorithm with pattern search for finding heavy atoms in protein crystals. ACM GECCO, p. 377–384.

Reeves, C. (1993). Using genetic algorithms with small populations. ICGA, p. 92–99.

Storn, R. and Price, H. (1997). Differential Evolution – A simple and efficient heuristic for global optimization over continuous spaces. J. Global Optimization, Vol. 11, p. 341–359.

Takagi, H. (2001). Interactive Evolutionary Computation: Fusion of the capabilities of ECO optimization and human evaluation. IEEE PIEEE, p. 1275–1296.

Tavares, J., Melab N. and Talbi, E. (2002). An Empirical Study on the Influence of Genetic Operators for Molecular Docking Optimization. PROTEINS: Structure, Function, and Genetics, Vol. 47: 409–443.

Thomsen, R. (2003). Flexible ligand docking using evolutionary algorithms: investigating the effects of variation operators and local search hybrids. Biosystems, Vol. 72: 57–73.

Thomsen, R. (2004). Multi-modal optimization using crowding-based differential evolution. IEEE CEC, p. 1382–1389.

Trott O., Olson A.J. (2009). Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading.Journal of Computational Chemistry.

DE code (2014)
http://www.icsi.berkeley.edu/~storn/code.html

CMA-ES code (2014)
https://www.lri.fr/~hansen/cmaes_inmatlab.html