SOLVING MOLECULAR DISTANCE GEOMETRY PROBLEMS BY GLOBAL OPTIMIZATION ALGORITHMS

ANDREA GROSSO, MARCO LOCATELLI, AND FABIO SCHOEN

Abstract. In this paper we consider global optimization algorithms based on multiple local searches for the Molecular Distance Geometry Problem (MDGP). Three distinct approaches (Multistart, Monotonic Bain Hopping, Population Basin Hopping) are presented and for each of them a computational analysis is performed. The results are also compared with those of two other approaches in the literature, the DGSOL approach [14] and a SDP based approach [1].

Keywords: Global Optimization, Distance Geometry, Multistart, Basin Hopping

1. Introduction

1.1. **Problem and paper contribution.** A distance geometry problem calls for determining positions for N points $x^1, x^2, \ldots, x^N \in \mathbb{R}^3$ such that, for a given set of pairs $\mathcal{D}$ and given bounds ℓ_{ij}, u_{ij}:

$$\ell_{ij} \leq \|x^i - x^j\| \leq u_{ij} \qquad \text{for all } \{i,j\} \in \mathcal{D}. \tag{1}$$

Distance geometry problems commonly arise in molecular chemistry, for determining the geometric structures of large molecules (e.g. proteins) from NMR experiments, which usually only deliver a subset of pairwise atom distances. In such contexts the problem is known as Molecular Distance Geometry Problem (MDGP) and each point in $\{1, \ldots, N\}$ represents (the center of) an atom; when $\ell_{ij} = u_{ij}$ for all $\{i,j\} \in \mathcal{D}$, i.e. all distances in $\mathcal{D}$ are known without errors, we call it *exact* MDGP.

The exact MDGP is polynomially solvable when distances $d_{ij} = \ell_{ij} = u_{ij}$ are known for all pairs with $i, j \in \{1, \ldots, N\}$, i.e. $\mathcal{D}$ includes all the atom pairs. Indeed, in this case a solution can be found by eigenvalue decomposition of the distance matrix, which requires $\mathcal{O}(N^3)$. But if not all distances are known the problem has been proved to be strongly NP-hard (see [13]).

The MDGP problem with uncomplete distance information has been tackled with a variety of techniques. In the EMBED algoritm (see [2]) first valid estimates of unknown distances are found, then a solution is obtained through eigenvalue decomposition of the distance matrix where the unknown distances are filled in with their estimates, and finally the obtained solution is refined by starting from it a local optimization with an appropriate objective function (for instance, function (2) below is suggested).

In [3] a build-up approach is used: a solution is progressively built up by fixing the unknown position of an atom on the basis of the known distances of this atom from other atoms whose position has been previously fixed.

A different approach proposed for the exact MDGP is proposed by Biswas et al. [1]. This determines an initial approximate solution X by solving a SDP relaxation, then performs standard local minimization from such a starting point. The use of decomposition techniques allows this approach to handle instances of

exact MDGP for quite large values of N (the idea of decomposing solutions was also exploited in the ABBIE approach [7]).

Some approaches rely on Global Optimization (GO) techniques; a correct molecular configuration $X = (x^1, x^2, \ldots, x^N) \in \mathbb{R}^{3N}$ must coincide with a global minimum of an error function

$$E_r(X) = \sum_{\{i,j\}\in\mathcal{D}} \left[\frac{\max^2\left(\ell_{ij}^2 - \|x^i - x^j\|^2, 0\right)}{\ell_{ij}^2} + \frac{\max^2\left(\|x^i - x^j\|^2 - u_{ij}^2, 0\right)}{u_{ij}^2} \right]. \tag{2}$$

Function $E_r(X)$ represents a total relative error computed for a configuration X with respect to constraints (1); similarly, one can use

$$E_a(X) = \sum_{\{i,j\}\in\mathcal{D}} \left[\max^2\left(\ell_{ij}^2 - \|x^i - x^j\|^2, 0\right) + \max^2\left(\|x^i - x^j\|^2 - u_{ij}^2, 0\right)\right], \tag{2'}$$

that represents a total absolute error. Clearly $E_r(X), E_a(X) \geq 0$ for any $X \in \mathbb{R}^{3N}$, and $E_r(X) = E_a(X) = 0$ if and only if X is a configuration satisfying (1). Although many error functions could be devised for testing the correctness of a configuration, E_a ad E_r are appealing for their smoothness that allows to use efficient local minimization procedures in search algorithms. Morè and Wu [14] propose the DGSOL algorithm for MDGP; DGSOL is based on *global continuation* techniques, and works by successive minimizations of smoothed functions $\langle E_r(X)\rangle_\lambda$, where $\langle E_r(X)\rangle_\lambda$ is obtained by convolution with a gaussian function $\lambda^{-N}\exp\left(-\|X\|^2/\lambda^2\right)$, for decreasing values of the parameter $\lambda \geq 0$ ($\langle E_r(X)\rangle_0$ is equivalent to E_r). In DGSOL $E_r(X)$ is smoothed to a convex (easy to minimize) function for large λ, whereas the original landscape is gradually restored as $\lambda \to 0$; as expected, stronger performances for DGSOL over simple multistart search are reported in [14]. Another GO method, GNOMAD [17], includes in the optimization problem domain knowledge through the addition of minimum separation distance constraints based on van der Waals interactions.

In this paper we present computational experiences on MDGP and exact MDGP with different GO algorithms based on multiple local searches. In Section 2 we discuss the well known Multistart algorithm. Although this is certainly not the most advisable GO algorithm for highly multimodal problems, our computational experience evidentiates two facts which is worthwhile to discuss. In Section 3 we discuss the Monotonic Basin Hopping (MBH) approach, a relatively simple but quite effective GO algorithm based on an appropriate choice of a neighborhood structure of local minima. In Section 4 we present Population Basin Hopping (PBH), a population-based variant of MBH. Finally, in Section 5 we make a computational comparison of PBH with the GO method DGSOL presented in [14] and with the SDP based method presented in [1].

All computational tests have been performed on an Opteron 252 processor 2.5GHz with 4GB RAM running Linux, and the algorithms have been implemented in C and C++.

Before starting the description and computational analysis of the algorithms, we give below a short description of the test instances used throughout the paper.

1.2. **Test cases.** All test instances in the paper have been generated from data in the Protein Data-Bank (PDB); we considered two classes of instances. First of all we adopted some instances from the work by Moré and Wu [14]; such instances have

Name	N	Density (%)
DNA200a	200	1.7
DNA200b	200	16.5
1PTQ	402	8.8
1HOE	558	6.5
1LFB	641	5.6
1PHT	814	5.3
1POA	914	4.1
1AX8	1003	3.7

TABLE 1. Instances used for testing. The Density parameter is $2|\mathcal{D}|/[N(N-1)]$.

been generated from the 1GPV molecule in PDB. Moré and Wu took 100-atom and 200-atom fragments, and generated distance data for pairs in successive residues R_k, R_{k+1}:

$$\mathcal{D} = \{\{i,j\} : i \in R_k, j \in R_{k+1}\},$$

setting the bounds to

$$\ell_{ij} = (1-\varepsilon)d_{ij}, \qquad u_{ij} = (1+\varepsilon)d_{ij}, \qquad \{i,j\} \in \mathcal{D},$$

with ε set to values from 0.04 to 0.16, with $\varepsilon = 0.04$ yielding the hardest instances. We considered two 200-atom instances — here called DNA200a and DNA200b — with $\varepsilon = 0.04$ provided with the DGSOL package.

We then generated instances of MDGP by a technique similar to that used in [1] for the exact MDGP. We selected a number of molecules from PDB for various sizes N and for each of them we kept in $\mathcal{D}$ only the pairs with interatomic distances d_{ij} below a cutoff value $R = 6\mathring{A}$; then we set

$$\ell_{ij} = (1-\varepsilon)d_{ij}, \qquad u_{ij} = (1+\varepsilon)d_{ij}, \qquad \{i,j\} \in \mathcal{D}$$

with ε randomly set in the interval $(0, 0.04)$. The choice of the interval is related to the observations given in [14], where bounds $\ell_{ij} = (1-0.04)d_{ij}$, $u_{ij} = (1+0.04)d_{ij}$ are claimed to give difficult instances — indeed, computational experience not discussed in the remainder of the paper confirms that for larger ε the resulting instances are quite easy.

Table 1 summarizes the characteristics of the tested instances.

2. Multistart approaches

Multistart is the simplest GO method among those based on multiple local searches. It merely consists in: randomly generating points within a domain D which is guaranteed to contain the global minimum; starting a local search from each of them; returning the detected local minimum with lowest function value. Multistart is advisable only for simple GO problems with few local minima and/or a large basin of attraction of the global minimum. This is usually not the case for distance geometry problems but we decided to present some results obtained with Multistart because they allow to make two interesting empirical observations.

The first observation is the unexpected dependence of Multistart's performance on the domain over which starting points for local searches are sampled. We restricted our attention to simple domains, namely $3N$-dimensional boxes, i.e. $D = [-R, R]^{3N}$. From the theoretical point of view, any box large enough to contain the global minimum is appropriate. From the set of known distances it is possible to estimate the diameter of the molecule and, therefore, to define a sufficiently large box containing it. For our instances choosing $R = 50$ is enough.

However, the remarkable empirical observation is that the size of the box is a key parameter for Multistart. In the first and second column of Table 2 we report the number of times the global minimum of function (2′) has been reached when starting local searches from 1000 random starting points sampled over a box with $R = 50$ and $R = 5000$ respectively. It is clear from the results that sampling on a larger box is (considerably) more efficient. What is going on here is probably something similar to what was observed in [8] for so called Big-Bang algorithm for Lennard-Jones clusters. That algorithm was simply based on generating an initial configuration of atoms in a Lennard-Jones cluster in a very narrow region and then starting a local search from such configuration. Within the very narrow region strong *repulsive* forces came into play spreading the atoms during the local search. It was observed that the detection of (putative) global minima was much easier in this way rather than when generating the initial configuration in a larger region. In our problem when generating over a very large region, function (2′) guarantees that strong *attractive* forces come into play. The obtained results suggest that these attractive forces are able to drive the local search procedure (limited memory BFGS in our tests) in such a way that it jumps over local minima with high function values and reaches more easily local minima with low function value (if not even the global minimum).

For the sake of completeness, we also tested as a possible generator of starting points for local searches, the random mechanism employed with the DGSOL algorithm. The results are reported in the third column of Table 2. We notice that these are not significantly different from those obtained by randomly generating over the box with $R = 50$, but still clearly worse than those obtained by random generation over the box with $R = 5000$.

The second interesting empirical observation is related to the choice of the objective function. From the theoretical point of view there is no difference between function (2) and function (2′): in both cases a global minimum with function value equal to 0 is a solution of our problem. However, there are clear differences from the computational point of view, a fact that, to the authors' knowledge, was not previously observed in the literature. In the fourth column of Table 2 we report the results for Multistart with random generation over a box with $R = 5000$ and objective function (2). These are clearly inferior with respect to the corresponding ones in the second column of the same table, obtained with objective function (2′). We point out that the superiority of function (2′) with respect to function (2) has not only been observed with the Multistart algorithm but also with all the other methods tested in this paper. A tentative explanation of this fact is the following. Assume the target distance for an atom pair is 0.5 and the current configuration violates it by 0.1, while the target distance for another atom pair is 5 and the current configuration violates it by 1. The correction of the error for the second atom pair requires a deeper modification of the current configuration with respect to the correction of the error for the first atom pair in view of the much larger *absolute* value of the error. But this fact does not emerge in (2) because the *relative* value of the error is exactly the same for both atom pairs.

3. Monotonic Basin Hopping

Key concepts in the literature about molecular conformation problems are those of *funnel* and *funnel bottom*, which we briefly review here (for more details see [11, 15, 16]). Let f be a function with a large number of local minima. Given a neighborhood structure $\mathcal{N}$ for the local minima of f, a funnel $\mathcal{F}$ can be defined as a maximal set of local minima such that for each $X \in \mathcal{F}$ there exists at least one

TABLE 2. Number of times Multistart has reached the global minimum over 1000 local searches with function (2′) and $R = 50$ (Column `AbsR50`), function (2′) and $R = 5000$ (Column `AbsR5000`), function (2′) and DGSOL initialization (Column `AbsDGSOL`), and function (2) and $R = 5000$ (Column`RelR5000`).

Instance	AbsR50	AbsR5000	AbsDGSOL	RelR5000
DNA200a	1	4	3	0
DNA200b	4	29	8	1
PTQ	1	49	0	0
HOE	0	23	2	2
LFB	0	0	0	0
PHT	0	2	0	0
POA	0	1	0	0
AX8	0	0	0	0
GPV	0	0	0	0

decreasing sequence

$$X_0 = X \to X_1 \to \cdots \to X_t = X^*, \quad f(X_i) < f(X_{i-1}) \text{ for all } i = 1, \ldots, t$$

of neighbor local minima (i.e. $X_i \in \mathcal{N}(X_{i-1})$ for all $i = 1, \ldots, t$) starting at X and ending at a common local minimum X^*, called the funnel bottom. Note that the global minimum is always a funnel bottom, independently of the neighborhood structure $\mathcal{N}$. The neighborhood must be of manageable size (in particular, much smaller than the whole set of local minima). In the easiest cases function f only has a single funnel, whose funnel bottom is the global minimum, in spite of its huge number of local minima. In these cases a single run of an algorithm which is able to detect funnel bottoms is also able to detect the global minimum. Here we describe a very simple, but at the same time very effective, algorithm to detect funnel bottoms. This is a slight generalization of the Monotonic Basin Hopping (MBH) algorithm (succesfully employed in [9] for the optimization of Lennard-Jones clusters), which in turn is a special case of the Basin Hopping algorithm [15]. The key operation of this algorithm is the *local move* Φ which, for a given local minimum X, returns a local minimum in its neighborhood, i.e.

$$\Phi(X) \in \mathcal{N}(X).$$

The choice of the local move and, therefore, also of the neighborhood explored by it, is essential for the performance of MBH. An indicative rule for its choice is that the local move should generate a new local minimum but without completely disrupting the structure of the current one X (the principle is very similar to that of reactive local searches in combinatorial optimization problems). Given the local move Φ, the MBH algorithm is defined as follows.

MBH(Φ):

Step 0 - Initialization: Randomly generate an initial local minimum X_0 and set $k = 0$.

Step 1- Child generation: Generate a child Y_{k+1} of X_k by the local move Φ, i.e. $Y_{k+1} = \Phi(X_k)$.

Step 2- Competition: Set:

$$X_{k+1} = \begin{cases} Y_{k+1} & \text{if } f(Y_{k+1}) < f(X_k) \\ X_k & \text{otherwise} \end{cases}$$

i.e. Y_{k+1} replaces X_k only if it has a better function value.

Step 3 - Stopping rule: Stop if either a global minimum is detected (this, of course, can usually only be checked if the global minimum value is known, which is the case for MDGP) or $k \geq$ `MAXITER`. Otherwise set $k = k+1$ and go back to Step 1.

In case the function has more than one funnel bottom it may be necessary to run MBH more than once (in a Multistart fashion) before detecting the global minimum.

For the distance geometry problem we propose two different local moves (although others can be thought of). One move is quite general, the other more problem-specific.

3.1. **Local move $\Phi_{1,\Delta}$.** The first local move that we tested is an extremely simple one and is basically the same employed for molecular conformation problems (see [4, 9, 10, 15]). Given a local minimum X, a local search procedure $\mathcal{L}$ and some $\Delta > 0$, we set

$$\Phi_{1,\Delta}(X) = \mathcal{L}(X + \delta),$$

where δ is randomly sampled within the box $[-\Delta, \Delta]^{3N}$. Basically, we randomly perturb the current local minimum, by controlling the size of the perturbation through Δ, and start a local search from the perturbed point.

The choice of the parameter Δ leads to an interesting empirical observation. As previously pointed out, the new local minimum generated by the local move should have a structure similar to the starting one. Therefore, we should avoid too large a value for Δ. On the other hand, we should also avoid too small a value to avoid that the perturbed point still lies in the region of attraction of the current local minimum and the local move does not move at all. What was experimentally observed is that Δ has to be chosen larger than one would expect, taking into account that the expected diameter of the solution in all our test instances never exceeds 50. The choice $\Delta = 5$ turns out to be too small (very often the local move was unable to escape from the current local minimum). Then, we increased it and found that $\Delta = 50$ was an appropriate choice, which was somehow surprising because we believed this was too large a value. To better understand what is going on, we collected more information. In particular, we produced the histograms of the distances between corresponding atoms (after appropriate rotations and translations) in X and $\Phi_{1,\Delta}(X)$. We could observe that, in spite of the large perturbation size $\Delta = 50$, the local search procedure is able to drive back the perturbed point towards a local minimum close to the original one. Indeed, the histograms were often concentrated around small distances.

We also tested the even larger perturbation size $\Delta = 500$. In this case the histograms were usually not concentrated around small distances but more spread, confirming that in these cases there is only a mild relation between X and its child $\Phi_{1,\Delta}(X)$ (the situation is rather similar to a pure Multistart approach with random generation over a large box).

Finally, we remark one possible limit of the above local move. Although the choice $\Delta = 50$ turned out to be a good one for all our test instances, more generally a correct choice may depend on the problem at hand, so that it may be necessary to introduce some adaptive strategy to set this value.

3.2. **Local move $\Phi_{2,\Gamma}$.** The second local move is problem-specific. In order to define it, we first need to define a local minimum Z_k as follows:

$$Z_k = \arg\max\{f(X_{k-1}), f(Y_k)\} \tag{3}$$

i.e. Z_k is equal to Y_k, the child of X_{k-1}, if Y_k is not accepted as the next incumbent at iteration k, otherwise it is equal to X_{k-1}. For each $r = 1, \ldots, N$, let:

$$\eta(r) = \max_{s:(r,s)\in\mathcal{D}} \{0, \|x_k^r - z_k^s\| - u_{rs}, \ell_{rs} - \|x_k^r - z_k^s\|\},$$

and let

$$s(r) \in \arg\max_{s:(r,s)\in\mathcal{D}} \{0, \|x_k^r - z_k^s\| - u_{rs}, \ell_{rs} - \|x_k^r - z_k^s\|\}.$$

Then we set

$$d_k^r = \begin{cases} 0 & \text{if } \eta(r) = 0 \\ -(x_k^r - z_k^{s(r)}) & \text{if } \eta(r) = \|x_k^r - z_k^{s(r)}\| - u_{rs(r)} \\ (x_k^r - z_k^{s(r)}) & \text{otherwise} \end{cases}$$

and $D_k = (d_k^1, \ldots, d_k^N)$. Finally, we define

$$\Phi_{2,\Gamma}(X) = \mathcal{L}(X + \gamma D_k),$$

where γ is randomly sampled in the interval $[0, \Gamma]$ (in our tests we fixed $\Gamma = 4$). Therefore, here the perturbation is defined through the collaboration of the current incumbent X_k with the previously discarded local minimum (either X_{k-1} or Y_k). If for some atom r the subset of distances in $\mathcal{D}$ involving these atoms are all within the prescribed bounds even when taking atom r in X_k and all the other atoms in Z_k, then we do not move this atom at all. Otherwise, if some of the bounds are violated we consider atom $s(r)$ in Z_k whose distance from atom r in X_k gives the largest violation with respect to the prescribed bounds and move atom r in X_k towards the position of atom $s(r)$ in Z_k if the upper bound is violated, or push it away from the position of atom $s(r)$ in Z_k if the lower bound is violated.

We remark that a reasonable alternative to the move proposed above could be to choose the direction of the displacement for atom r only on the basis of internal bound violations of the current incumbent, i.e. the same as above but with Z_k substituted by X_k itself. Unfortunately, our experiments with this alternative move did not give as good results as those obtained with $\Phi_{2,\Gamma}$.

3.3. **Computational results.** The computational results of MBH with the two local moves described above ($\Phi_{1,\Delta}$ with $\Delta = 50$, and $\Phi_{2,\Gamma}$ with $\Gamma = 4$) over our set of instances are reported in Table 3. We notice that in both cases the percentage of successes is quite high, suggesting that these problems have just one or, at most, few funnel bottoms with respect to the neighborhoods corresponding to the proposed local moves. Therefore, few runs of MBH are enough to detect the global minimum. Columns `SUCC` (number of successes over 10 runs) in Table 3 suggest that the problem-specific local move $\Phi_{2,\Gamma}$ reaches more easily a funnel bottom not corresponding to the global minimum with respect to the more general local move $\Phi_{1,\Delta}$. This makes the overall effort per success with $\Phi_{2,\Gamma}$ usually larger than that with $\Phi_{1,\Delta}$. On the other hand, columns `NSavg-succ` (average number of local searches *only in the successful runs*) suggest that, when starting in the funnel whose funnel bottom is the global minimum, local move $\Phi_{2,\Gamma}$ usually reaches the global minimum faster than local move $\Phi_{1,\Delta}$.

4. Population Basin Hopping

The results for MBH show that it would be important to recognize as soon as possible when a run of MBH leads to a failure. Especially for $\Phi_{2,\Gamma}$, the `NSavg-succ` values in Table 3, i.e. the average number of local searches in successful runs, is usually much lower than the overall average number of local searches per success, i.e. the values `NSavg` in Table 3. The latter number is made larger by the `MAXITER` local searches (500 in our tests) which have to be added each time a failure occurs.

TABLE 3. Number of successes (columns SUCC) over 10 runs, overall average number of local searches per success (columns NSavg), and average number of local searches in successful runs (columns NSavg-succ) for MBH with the two local moves $\Phi_{1,\Delta}$ and $\Phi_{2,\Gamma}$.

	$\Phi_{1,\Delta}$			$\Phi_{2,\Gamma}$		
Instance	Succ	NSavg	NSavg-succ	Succ	NSavg	NSavg-succ
DNA200a	10	151	151	10	39	39
DNA200b	10	24	24	10	20	20
PTQ	10	15	15	7	217	3
HOE	10	17	17	8	139	14
LFB	10	55	55	8	192	67
PHT	10	55	55	8	136	11
POA	9	180	124	8	152	27
AX8	10	107	107	9	74	18
GPV	4	888	138	6	470	137

Obviously, if we could choose MAXITER as low as possible, i.e. if we could stop a run as soon as possible, without reducing the number of successes, we could reduce the overall effort per success. Unfortunately, this is not an easy task and, as suggested by the large variability of the NSavg-succ values in Table 3, the appropriate value is quite different from instance to instance.

A simple way to overcome this difficulty is to follow in parallel K different trajectories. Of course, this increases the effort per iteration (by a factor of K) but we increase the probability of success and, above all, the total number of iterations to reach the global minimum is determined by the shortest of the K trajectories, which partially counterbalances the larger effort per iteration. Note that we still have to choose a parameter (the number K of trajectories) but this is less variable from instance to instance (in our tests we always fixed $K = 10$).

The easiest way to follow K trajectories is to run K parallel independent runs of MBH, i.e. to follow K independent trajectories. However, here we present a more general algorithm where trajectories are not necessarily independent. This is the PBH algorithm already tested in [5] on cluster optimization problems. The main elements of this algorithm are a local move Φ as in MBH, and a dissimilarity measure d between local minima.

PBH(Φ, d)**:**

Step 0 - Initialization: Create an initial random population of K individuals

$$\mathcal{X}_0 = \{X_0^1, \ldots, X_0^K\}$$

and set $k = 0$.

Step 1- Children generation: Generate a set of K children of the members of the population

$$\mathcal{Y}_{k+1} = \{Y_{k+1}^1, \ldots, Y_{k+1}^K\},$$

where $Y_{k+1}^j = \Phi(X_k^j)$ for all $j = 1, \ldots, K$.

Step 2 - Competitor selection: For each child Y_{k+1}^j select its competitor $X_k^{i(j)}$ within $\mathcal{X}$ according to the following rule:

$$i(j) \in \arg\min_{t=1,\ldots,K} d(Y_{k+1}^j, X_k^t),$$

i.e. the competitor is the member of the population less dissimilar from Y_{k+1}^j.

Step 3- Competition: For each j, in the new population we will have

$$X_{k+1}^{i(j)} = \arg\min\{f(Y_{k+1}^{j}), f(X_{k}^{i(j)})\}$$

Step 4 - Stopping rule: Stop if either a global minimum is detected (local minimum with value 0) or $k \geq$ `MAXITER`. Otherwise set $k = k + 1$ and go back to Step 1.

PBH includes as a special case the one of K independent trajectories: this occurs when the competitor of a child Y_{k+1}^{j} is always its own father X_{k}^{j}, i.e. when $i(j) \equiv j, \ \ \forall\ j$. When this is not the case, there is *communication* and *collaboration* between members of the population controlled by the dissimilarity measure. If the dissimilarity measure is appropriately chosen, it allows to counterbalance the too greedy tendency of MBH (too fast convergence to a funnel bottom not corresponding to a global minimum) leading to a much greater efficiency with respect to the case of K independent trajectories, as observed in [5] and in the related experimental analysis in [6]. This usually happens when many funnel bottoms exist and/or the funnel bottom corresponding to the global minimum can be reached through trajectories of considerably different lengths. It is still not clear to us whether this is the case for distance geometry problems. After testing a few dissimilarity measures, we restricted our attention to a very general and simple one, the absolute value of the difference between function values:

$$d(X, Y) = \mid f(X) - f(Y) \mid . \tag{4}$$

The results obtained with this measure (reported in Table 4) are good ones but basically comparable with those obtained with independent trajectories. Therefore, at the moment we can not claim that for this problem the use of dissimilarity measures considerably increases the efficiency (as it happens for other problems, see [5]). In spite of this, we decided to present in this paper the PBH setting in the hope that future researches will discover new, problem-specific dissimilarity measures which do allow to increase the efficiency (see also the discussion at the end of this section about a positive effect of collaboration and communication between members of the population).

Before presenting the results, we just remark a small difference in the definition of Z_k with respect to (3) for the local move $\Phi_{2,\Gamma}$. In PBH we define a point Z_k^j for each member j of the population as follows:

$$Z_{k}^{j} = \arg\max\{f(X_{k-1}^{i(j)}), f(Y_{k}^{j})\}$$

The results with both local moves $\Phi_{1,\Delta}$ and $\Phi_{2,\Gamma}$ are reported in Table 4. In both cases the population is large enough to guarantee 100% of successes on all test instances. Moreover, as expected from the values in Columns `NSavg-succ` in Table 3, we observe that better results are usually obtained with the problem-specific move $\Phi_{2,\Gamma}$ with respect to $\Phi_{1,\Delta}$.

We conclude this section by mentioning some experiments which allows to appreciate the positive effects of the dissimilarity measure. We tested local move $\Phi_{2,\Gamma}$ with $\Gamma = 2$. This choice for Γ turned out to be too small: the corresponding neighborhood structure is such that too many funnel bottoms are present. This lead to many failures for MBH and also for PBH with $K = 10$. But in this situation the effect of the population became clear when we enlarged the population size to $K = 40$. We immediately reached 100% of successes with this choice. When looking into these results we realized that these were mostly due to the heavy influence of *survival*, one of the two phenomena (the other is *backtracking*) which, according to the analysis in [5, 6], is responsible for the successes of PBH. Survival occurs when a member X_{k}^{j} of the population generates a better child Y_{k+1}^{j}, i.e.

TABLE 4. Number of successes (columns SUCC)) over 10 runs and average number of local searches per success (columns NSavg) for PBH with the two local moves $\Phi_{1,\Delta}$ and $\Phi_{2,\Gamma}$.

	$\Phi_{1,\Delta}$		$\Phi_{2,\Gamma}$	
Instance	Succ	NSavg	Succ	NSavg
DNA200a	10	116	10	143
DNA200b	10	30	10	23
PTQ	10	57	10	33
HOE	10	67	10	53
LFB	10	192	10	89
PHT	10	162	10	63
POA	10	234	10	88
AX8	10	263	10	87
GPV	10	949	10	312

$f(Y^j_{k+1}) < f(X^j_k)$, but $i(j) \neq j$ so that X^j_k will still survive in the next population. This phenomenon (through which communication and collaboration between members of the population takes place) counterbalances the greedy tendency of MBH, which might cause too fast convergence to funnel bottoms not corresponding to the global minimum. Indeed, it allows to keep in the population local minima from which it is still possible to reach the global minimum.

Overall, the performance of PBH with $\Gamma = 2$ and $K = 40$ is inferior to that with $\Gamma = 4$ and $K = 10$, but still it is interesting to observe how the collaboration and communication between members of the population is able to considerably reduce the negative effects of a wrong choice of the parameter Γ defining the local move.

5. Comparison with existing approaches

We now compare the PBH approach with other existing algorithms. For the general (not exact) MDGP problem, we consider DGSOL a state of the art algorithm. Table 5 compares the results delivered by PBH (with the $\Phi_{2,\Gamma}$ local move) with those delivered by DGSOL, in terms of number of successes (over 10 runs) and average CPU times in seconds. DGSOL is quite efficient on the DNA* instances, still PBH is able to outperform it (except for the CPU time on DNA200a). Also, the larger 1PTQ,..., 1AX8 strongly defeat DGSOL; we can draw some hypotheses in order to address this behaviour.

- First of all 1PTQ,...,1AX8 are quite large instances; at the time of writing we are not aware of literature about testing of DGSOL on such large instances.
- The method used for generating the large instances is different from that used in [14]; this could result in a somehow harder class of instances.
- Also, an important phenomenon has to be mentioned. For some instances, we fed DGSOL with an initial solution quite close to the optimal solution; then the smoothing and search phase delivered a non-optimal solution quite different from the optimal configuration. This may indicate that in such cases the smoothed, convex version of the objective function — $\langle E_r \rangle_\lambda$ for large λ — may have a global minimum far from the global minimum of E_r, and thus the landscape of $\langle E_r \rangle_\lambda$ is poorly related to that of E_r, and successive optimizations with decreasing λ are not always able to drive the search towards the correct point.

	PBH $\Phi_{2,\Gamma}$		DGSOL	
Instance	Succ	CPU	Succ	CPU
DNA200a	10	85	5	47
DNA200b	10	12	7	31
1PTQ	10	11	0	–
1HOE	10	20	0	–
1LFB	10	99	0	–
1PHT	10	111	0	–
1POA	10	92	0	–
1AX8	10	110	1	820
1GPV	10	1983	0	–

TABLE 5. Number of successes over 10 runs (Columns SUCC) and average CPU times in seconds (Columns CPU), PBH vs DGSOL.

For the exact MDGP, leading results are reported for the SDP-based techinque by Biswas et al. in [1]. We generated instances from molecules 1PTQ,...,1AX8 by retaining exact information on the distances d_{ij} for the pairs in $\mathcal{D} = \{\{i,j\} : d_{ij} \leq R\}$, with $R = 6\mathring{A}$. According to Table 6, the PBH approach is able to find the global minima for the tested instances; a percentage of 10/10 successes is reported in all cases. Comparing CPU times in a meaningful way is difficult since the CPU times in the SDP column are taken from [1], and were obtained on a different processor and with different environments (i.e., MATLAB). While our processor is about 1.2 times faster than that used in the cited work, it should still be remarked that MATLAB software may be (sometimes significantly) slower than optimized C++ code. The SDP-based approach has not been extended to the cases of non-exact distances in [1], apparently because the involved SDP model seems to be specifically tailored to the exact case. The authors report about some experiment with *noisy* data, but no attempt is made of dealing with upper and lower bounds on distances. Besides, the PBH approach seems to be reasonably well-suited for both kinds of problem; the general MDGP requires more CPU time than the exact version, but the worsening of performances is not dramatic.

With respect to the correct reconstruction of the molecular configuration corresponding to each test instance, common measures of performance are the minimum root mean-square deviation (RMSD) between the actual and estimated coordinates and the distance mean error with respect to all distances (DME) and with respect to the given distances (LDME). Such measures are defined by

$$\text{RMSD} = n^{-1/2} \min_{Q} \left\{ \left[\sum_{i=1}^{N} \|x^i - Q\bar{x}^i\|^2 \right]^{1/2} \right\},$$

where $\bar{x}^1, \ldots, \bar{x}^N$ are the actual coordinates of each atom (both X and $\bar{X}$ configurations are assumed to be centered at the origin), and Q is a 3×3 orthogonal matrix;

$$\text{DME} = \left[\frac{2}{n(n-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \left(\|x^i - x^j\| - \|\bar{x}^i - \bar{x}^j\| \right)^2 \right]^{1/2},$$

$$\text{LDME} = \left[\frac{1}{|\mathcal{D}|} \sum_{\{i,j\} \in \mathcal{D}} \left(\|x^i - x^j\| - \|\bar{x}^i - \bar{x}^j\| \right)^2 \right]^{1/2},$$

For all such measures, PBH was able to deliver results for the exact-MDGP with small deviations; the results are reported in Table 7.

TABLE 6. Average CPU times in seconds over the six instances with up to 1000 atoms from [1] for PBH with local move $\Phi_{2,\Gamma}$, dissimilarity measure (4) and $K = 10$ both with bound on distances (Column `Bound`) and with exact distances (Column `Exact`), and CPU times in seconds for the SDP approach presented in [1] with exact distances (Column `SDP`).

	PBH		
Instance	Bound	Exact	SDP
PTQ	11	4	121
HOE	19	9	319
LFB	98	19	195
PHT	111	68	461
POA	92	35	361
AX8	109	28	493

TABLE 7. Error measures for exact-MDGP tests.

Instance	RMSD ($\times 10^{-6}$Å)	DME ($\times 10^{-6}$Å)	LDME ($\times 10^{-6}$Å)
PTQ	2.65	1.85	$4.73 \cdot 10^{-2}$
HOE	2.21	1.69	$5.73 \cdot 10^{-2}$
LFB	3.03	2.22	$5.80 \cdot 10^{-2}$
PHT	28.62	29.50	$7.69 \cdot 10^{-2}$
POA	32.63	64.50	$7.28 \cdot 10^{-2}$
AX8	2.40	1.79	$7.70 \cdot 10^{-2}$
GPV	916.92	499.34	$1.16 \cdot 10^{-1}$

REFERENCES

[1] Biswas P., Liang T.-C., Toh K.-C., Ye Y., An SDP based approach for anchor-free 3D graph realization, available at `www.stanford.edu/~yyye/distmolecule6.pdf` (2006)

[2] Crippen G., Havel T. , *Distance geometry and molecular conformation*, Wiley (1988)

[3] Dong Q., Wu Z., A geometric build-up algorithm for solving the molecular distance geometry problem with sparse distance data, *J. Global Opt.*, 26, 321-333 (2003)

[4] J. P. K. Doye, R. H.Leary, M. Locatelli and F. Schoen, The global optimization of Morse clusters by potential transformations, *INFORMS J. Computing*, 16, 371-379, (2004).

[5] Grosso A., Locatelli M., Schoen F., A population based approach for hard global optimization problems based on dissimilarity measures, to appear in *Mathematical Programming*, (2006)

[6] Grosso A., Locatelli M., Schoen F., An experimental analysis of population based approach for global optimization, to appear in *Comp. Opt. Appl.*. (2006).

[7] Hendrickson B.A., *The molecular problem: determining conformation from pairwise distances*, Ph. D. Thesis, Cornell University (1991)

[8] R. H. Leary, Global optima of Lennard-Jones clusters, *Journal of Global Optimization*, 11, 35-53 (1997)

[9] R. H. Leary, Global optimization on funneling landscapes, *J. Global Optim.*, 18, 367–383, (2000).

[10] M.Locatelli, F.Schoen, Efficient algorithms for large scale global optimization: Lennard-Jones clusters, *Computational Optimization and Applications*, 26, 173-190 (2003).

[11] M.Locatelli, On the multilevel structure of global optimization problems, *Computational Optimization and Applications*, 30, 5-22 (2005)

[12] J. Moré, Z. Wu, Global continuation for distance geometry problems, *SIAM J. Optim*, 7, 814-836 (1997)

[13] Saxe J.B., Embeddability of graphs in $k-$space is strongly NP-hard, in *17th Allerton Conference in Communication, Control and Computing*, 480-489 (1979)

[14] J. Moré, Z. Wu, Distance geometry optimization for protein structures, *Journal of Global Optimization*, 15, 219–234 (1999).

[15] D. J. Wales and J. P. K. Doye, Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms, *J. Phys. Chem. A*, 101, 5111–5116, (1997).

[16] D. J. Wales, *Energy Landscapes with Applications to Clusters, Biomolecules and Glasses*, Cambridge University Press, Cambridge, (2003).

[17] G. A. Williams, J. M. Dugan, R. B. Altman, Constrained global optimization for estimating molecular structure from atomic distances, *Journal of Computational Biology*, 8, 523–547 (2001)

Dip. Informatica - Università di Torino (Italy)
E-mail address: grosso@di.unito.it

Dip. Informatica - Università di Torino (Italy)
E-mail address: locatell@di.unito.it

Dip. Sistemi e Informatica - Università di Firenze (Italy)
E-mail address: schoen@ing.unifi.it