Journal of Informatics and Mathematical Sciences
Volume 5 (2013), Number 2, pp. 77–100
RGN Publications

http://www.rgnpublications.com

Algorithms for Constrained Best-fit Alignment

Laure Devendeville, Serge Dumont, Olivier Goubet, and Sylvain Lefebvre

Abstract Manufacturing complex structures as planes requires the assembly of several pieces. The first step in the process is to align the pieces. This article is concerned with some mathematical and computational aspects of new algorithms devoted to the alignment of the pieces. We describe the properties of suitable algorithms to handle a non standard constrained optimization problem that occurs in the assembly process of a manufactured product. Then we present two kinds of algorithms: the first based on a fractional step algorithm and the second on a local search algorithm. We assess them on real cases and compare their results with an evolutionary algorithm for difficult non-linear or non-convex optimization problems in continuous domain.

1. Introduction

1.1. *Setting the problem*

Manufacturing complex structures as planes requires the assembly of several pieces. The first step in the process is to align the pieces. The assembly process should respect some functional geometric requirements. Actually, it is impossible to product pieces that allow a hole-to-hole (or precise) assembly. Real workpieces are rigid systems that come along with tolerances with respect to perfect nominal pieces. In the last decade, a new assembly process has been introduced in aeronautics: the Best-Fit process that can be described as follows. First, take some measurements, with some tracking lasers, on the pieces to be assembled. Then compute with a suitable software the displacement of the pieces that align the pieces accordingly to the tolerances. Then perform the assembly.

This article is concerned with some mathematical and computational aspects of new algorithms devoted to the alignment of the pieces. This work was initiated

2010 *Mathematics Subject Classification.* 65K05, 90C55, 90C15, 90C26.
Key words and phrases. Non convex constrained optimization problems; Steepest descent with projection algorithm and stochastic local search algorithm.
This work was supported by the HTSC program Tolérants of the *Contrat de plan État-Région Picardie*.

to answer a request from the European Aeronautic Defence and Space Company (EADS-IW). Other applications to real industrial problems of the algorithm will be developed elsewhere.

1.2. *Outline of the article*

In a second section we describe the mathematical framework of the problem to handle. The tolerances provide us with constraints. Therefore we are given a constrained optimization problem. We then translate these optimization problem into mathematical language by introducing some cost or fitness functional that may differ according to the target we want to reach. Section 3 is devoted to introduce a fractional step algorithm to solve the optimization problem. This provide us with an efficient method to compute a solution. The difficulty is that our algorithm is a deterministic algorithm that applies to a non convex functional; therefore some improvements are required to avoid to be trapped in the neighborhood of a local minimizer. Therefore in section 4 we introduce a greedy stochastic algorithm, based on the previous one, to improve the results of the previous section. In section 5 we present CMA-ES ([8]) that is an evolutionary algorithm for difficult non-linear or non-convex optimization problems in continuous domain. Finally in section 6 we discuss the efficiency of these algorithms on both academic and real benchmarks. A last section contains some conclusions and perspectives.

2. The Problem

2.1. *Mathematical modeling*

Consider two rigid bodies $\mathscr{V}, \mathscr{T}$ in $\mathbb{R}^3$. On each body is given (or measured) a set of points denoted respectively $\{v_j\}_{1\leq j\leq n}$, $\{t_j\}_{1\leq j\leq n}$. Here we are dealing with a $3D$ problem; it is worth to point out that if some symmetries are valid, one can assume that the sets are included in a plane or in a line, and that some other specific algorithms for $1D$ or $2D$ problems can be used. This will not be developed here (we refer to [13] and to the references therein).

A hole-to-hole (or perfect) assembly requires to compute a displacement D such that $Dv_j = t_j$ for all j. In fact, some tolerances are allowed to relax these constraints. These tolerances are constraints that are expressed as follow. For each j, let be given a convex closed subset $\bar{\Omega}_j$ whose interior Ω_j is non empty (without loss of generality we may assume that 0 belongs to this interior). The tolerance will be satisfied if we are able to find out a displacement D such that

$$Dv_j - t_j \in \bar{\Omega}_j, \quad \text{for each } j. \tag{2.1}$$

The ultimate goal of the assembly process is to find out a displacement such that the above assertions hold true for any j. For real life problems in aeronautics, it can occur that such a solution does not exist, or that one cannot compute straightforwardly the displacement that allow the hole-to-hole assembly. Hence

we introduce a *fitness* function as the derogation number $N(D)$, that is the number of j such that (2.1) is not valid, i.e.

$$N(D) = \#\{j;\ Dv_j - t_j \notin \bar{\Omega}_j\}. \tag{2.2}$$

This derogation number is really a cost function because any non zero derogation number implies a fee that is a loss of money for the plane manufacturers.

Let now go back to the mathematical modeling. A displacement D acting on $\mathbb{R}^3$ is an affine isometry of $\mathbb{R}^3$. Let us recall that such a displacement is defined from a matrix M that belongs to the orthogonal group $O(3)$ and a translation τ through the following product: for any z in $\mathbb{R}^3$

$$Dz = Mz - \tau. \tag{2.3}$$

We now define the *energy* functional as

$$E(D) = \frac{1}{2}\sum_{j=1}^{n} |Dv_j - p_j(Dv_j)|^2, \tag{2.4}$$

where p_j denotes the projector onto the closed convex set $t_j + \bar{\Omega}_j$ and $|\cdot|$ is the Euclidean norm on $\mathbb{R}^3$; let us recall the very definition of this projector: for any $z \in \mathbb{R}^3$, $|z - p_j(z)| = \min_{v \in t_j + \bar{\Omega}_j} |z - v|$ (other choices for the projection are discussed in [**?**]). Any displacement such that $E(D) = 0$ avoid positive derogation number, and is then a solution to the original problem.

Let us observe that we have substituted to a discrete fitness function (2.2) an energy functional $E(D)$ that is defined on a continuous space, namely the set of affine isometries. The idea is then to process to minimize this energy functional $E(D)$. We introduce in the sequel a efficient algorithm to achieve this goal. In the next section, we will go further using this algorithm combined with some greedy stochastic procedure to minimize the fitness functional $N(D)$.

3. An Energy Decreasing Algorithm

The following difficulties occur about the optimization problem. On the one hand, the optimization problem is non convex since the set of affine isometries is non convex. Therefore classical algorithms do not apply straightforwardly. On the other hand, we do not know if either a solution does exist or conversely if there is uniqueness of such solution.

We introduce in the sequel a fractional step algorithm (or splitting algorithm) inspired by the algorithm that decreases the energy of a liquid crystal (see [1]). Let us remind that this algorithm was introduced to minimize a classical energy as $\int_\Omega |\nabla u|^2$ among smooth function u that satisfy $|u| = 1$ a.e. This last constraint is not convex. Then the fractional step algorithm reads as follows. Set u^0 for the initial guess. At each stage u^k for k integer preform two steps. Relax the constraint $|u| = 1$ and decrease the energy by a gradient method. Set $u^{k+\frac{1}{2}}$ for the point that achieves this descent. Then project $u^{k+\frac{1}{2}}$ into the set of functions satisfying the constraint.

Set u^{k+1} for the projection. Then go to the next stage. This algorithm converges due to the fact that the projection process decrease also the energy (see [1]).

Let us introduce now our algorithm.

- Initialize the algorithm by $D_0 = I$, the identity matrix for instance. At each stage k, D_k being provided, perform the following two steps
- *First step*: consider the projection $p_j(D_k v_j)$ of the points v_j onto the closed convex set $t_j + \bar{\Omega}_j$.
- *Second step*: compute D_{k+1} that minimizes

$$J(D) = \frac{1}{2}\sum_{j=1}^{n} |Dv_j - p_j(D_k v_j)|^2. \tag{3.1}$$

- Then go to the next stage.

The first step can be understood as follows: let us pretend that the body $\mathcal{V}$ is not rigid anymore. By the projection method, we find a displacement of this body that is not an isometry but that fits with the tolerances. This step is therefore called the *constrained* step. In the second step, we seek an isometry that is close (in the least-square sense) to the previous displacement. We call this step the *rigid* one. Throughout this article we shall refer to this as the Constrained/Rigid (CR) Algorithm.

3.1. *Statement of the main mathematical result*

We now state

Theorem 3.1. *The CR algorithm is well defined and provides us with a sequence D_k that decreases the energy. If moreover the matrix H_0 whose entries are $\sum_{j=1}^{n} (v_j)_l (t_j)_m$, $1 \le l,\ m \le 3$ is invertible and if the domains Ω_j are small enough, then the sequence D_k converges towards a limit D_∞.*

Proof. To begin with, up to a translation, we may assume that

$$\sum_{j=1}^{n} v_j = 0. \tag{3.2}$$

First step: An energy decreasing algorithm.

We now prove that there is one solution D that achieves the minimum of the functional (3.1) (we give in the sequel a process to compute it efficiently). For the sake of simplicity, set $c_j = p_j(D_k v_j)$. We seek D as M, τ that are respectively an element of $O(3)$ and a translation. Then, the functional (3.1) reads, denoting the scalar product on $\mathbb{R}^3$ by $(\cdot,\cdot)$,

$$J(D) = J(M,\tau) = \frac{1}{2}\sum_{j=1}^{n} |Mv_j|^2 - \sum_{j=1}^{n} (Mv_j, \tau + c_j) + \frac{1}{2}\sum_{j=1}^{n} |\tau + c_j|^2$$

$$= \frac{1}{2}\sum_{j=1}^{n}|v_j|^2 - \sum_{j=1}^{n}(Mv_j, c_j) + \frac{1}{2}\sum_{j=1}^{n}|\tau + c_j|^2, \tag{3.3}$$

since M is an isometry and since $\sum_{j=1}^{n} v_j = 0$. At this stage, the minimization problem splits into two minimization problems respectively in τ and in M. The first one admits as a solution

$$\tau = -\frac{1}{n}\sum_{j=1}^{n} c_j = -\frac{1}{n}\sum_{j=1}^{n} p_j(M_k v_j - \tau_k). \tag{3.4}$$

The second one reads: Find M in $O(3)$ that maximizes

$$G(M) = \sum_{j=1}^{n}(Mv_j, c_j) = \mathrm{Tr}(H^* M), \tag{3.5}$$

where $H_{l,m} = \sum_{j=1}^{n}(v_j)_l (c_j)_m$, for $1 \le j, l \le 3$. We use the following statement

Lemma 3.2 (Polar decomposition of a matrix). *For any matrix H there exists a pair (U, S) that belongs to $O(3) \times \mathcal{S}_3$, where $\mathcal{S}_3$ is the set of symmetric matrices with non negative eigenvalues such that $H = US$.*

At this stage, we seek an isometry N that maximizes $\mathrm{Tr}(SN)$, with $UN = M$. Computing in a basis where S is a diagonal matrix, we thus obtain

$$\mathrm{Tr}(SN) = \sum_{l=1}^{3} \lambda_l N_{ll}, \tag{3.6}$$

where $\lambda_1 \le \lambda_2 \le \lambda_3$ are the eigenvalues of S. We easily observe that this sum is maximal for $N = I$ and thus $U = M$.

At this stage we know that the algorithm is well defined, then we argue about its convergence properties. On the one hand, by the very definition of D_{k+1},

$$\frac{1}{2}\sum_{j=1}^{n}|D_{k+1}v_j - p_j(D_k v_j)|^2 \le \frac{1}{2}\sum_{j=1}^{n}|D_k v_j - p_j(D_k v_j)|^2 = E(D_k). \tag{3.7}$$

On the other hand, due to the projection property

$$|D_{k+1}v_j - p_j(D_{k+1}v_j)|^2 \le |D_{k+1}v_j - p_j(D_k v_j)|^2. \tag{3.8}$$

Therefore $E(D_{k+1}) \le E(D_k)$. Let us observe that if the equality $E(D_{k+1}) = E(D_k)$ is valid, then going back to the algorithm, it is straightforward to check that the sequence D_l is stationary for $l \ge k+1$.

Observe that the sequence τ_k is trapped into the compact convex set $K = \frac{1}{n}\sum_{j=1}^{n}\bar{\Omega}_j$. Then we have a sequence $D_k = (M_k, \tau_k)$ that belongs to the compact set $O(3) \times K$. There exists then a subsequence $D_{k'}$ that converges to D_∞. To prove that the whole sequence is convergent we need a technical assumption.

Second step: using the technical assumption.

Assume that H_0 is invertible. By a continuity argument for any sequence c_j that belong to $t_j + \overline{\Omega}_j$ satisfy that the matrix H whose entries are $\sum_{j=1}^{n}(v_j)_l(c_j)_m$ is also invertible; then this remains true for the particular choice $c_j = p_j(D_k v_j)$ throughout the algorithm process. As long as H is invertible, the map $f : D_k \mapsto D_{k+1}$ is continuous, since in the polar decomposition of a matrix the map $H \mapsto (U,S)$ is continuous.

Therefore if $D_{k'} \mapsto D_\infty$, then $D_{k'+1} \mapsto f(D_\infty)$. We are going to prove that $f(D_\infty) = D_\infty$ that will lead to the convergence of the whole sequence. We have

$$\begin{aligned} E(D_\infty) &= \lim E(D_{(k+1)'}) \le E(f(D_\infty)) \\ &= \lim E(D_{k'+1}) \le E(D_\infty) \\ &= \lim E(D_{k'}). \end{aligned} \tag{3.9}$$

Therefore $E(D_\infty) = E(f(D_\infty))$ and D_∞ is a fixed point of f. □

3.2. *Miscellaneous remarks and comments*

3.2.1. *About the technical assumption.* There is one ambiguity in the proof of the convergence of the algorithm. Imagine that there exists a stage with the matrix H that is not invertible. Then in the polar decomposition the matrix U is not unique (anyway, any choice of U will do the job). For real life applications in aeronautics, this cannot occur. In fact the measured points (t_j, v_j) are close to the perfect nominal points (T_j, V_j) given by the Computer Aid Design. These perfect points satisfy that there exists an exact displacement D such that $DV_j = T_j$. The corresponding perfect matrix H_{CAD} whose entries are $\sum_{j=1}^{n}(V_j)_l(T_j)_m$ is invertible; actually if $x \in \mathbb{R}^3$ is in the kernel of H_{CAD}, then for any vector $y \in \mathbb{R}^3$, $\sum_{j=1}^{n}(V_j, y)(T_j, x) = 0$. Choosing $y = D^*x$ leads to $(T_j, x) = 0$ for all j and then $x = 0$ if we have enough points T_j to span $\mathbb{R}^3$ (n is a large number in applications). Then by a continuity argument H_0 is also invertible.

3.2.2. *The CR algorithm applied to pure translations displacements.* We give a convergence result that is valid in a particular case. Assume for the sake of simplicity that the Ω_js are balls and that we seek a solution that is a translation, i.e. with $M = 0$. This strategy can be advocated as follows: the main part of the displacement is given by the translations, the rotation allowing some adjustment on the alignment. We now state

Proposition 3.3. *Minimizing*

$$E(\tau) = \frac{1}{2}\sum_{j=1}^{n}|v_j - \tau - t_j - p_j(v_j - \tau - t_j)|^2,$$

on the set of translations amounts to minimize a convex functional.

Proof. Set $z_j = v_j - t_j$. Set $E_j(\tau) = |z_j - \tau - p_j(z_j - \tau)|^2$. Assume that Ω_j is the ball centered at 0 and of radius δ_j. Then $\nabla E_j(\tau) = \left(\left(1 - \frac{\delta_j}{|z_j - \tau|}\right)_+\right)^2 |z_j - \tau|^2$, where $x_+ = \max(x, 0)$. Therefore one can prove that

$$E_j(\sigma) - E_j(\tau) - (\nabla E_j(\tau), \sigma - \tau) \geq 0. \tag{3.10}$$

Hence E is convex as the sum of convex functions. □

In this case the CR algorithm is actually a gradient algorithm, since the iteration process reduces to

$$\tau_{k+1} = \tau_k - \frac{1}{n} \sum_j \left(1 - \frac{\delta_j}{|z_j - \tau|}\right)_+ (z_j - \tau_k). \tag{3.11}$$

3.3. *An over-relaxed CR algorithm*

An over-relaxed version of the CR algorithm was also used for applications (see [13]). Let us describe in few words this over-relaxation process. Introduce a parameter $\lambda \in [0, 1]$. The idea is to enforce the constraints by computing the projections onto the sets $t_j + \lambda\overline{\Omega_j}$ instead of $t_j + \overline{\Omega_j}$.

The effect of this parameter transpires on the pure translation displacement case. If we are interested in computing the speed of convergence of this iteration process (3.11), using that p_j is a 1-Lipschitzian operator, we have that if $\varphi(\tau_k)$ denotes the right hand side of (3.11)

$$\begin{aligned} |\varphi(\tau) - \varphi(\sigma)| &\leq \frac{1}{n} \sum_{j; z_j - \tau \notin \Omega_j} |p_j(z_j - \tau) - p_j(\sigma_j - \tau)| \\ &\leq \frac{\#\{j;\ Dv_j - t_j \notin \bar{\Omega}_j\}}{n} |\tau - \sigma|. \end{aligned} \tag{3.12}$$

The rate of the geometric convergence, i.e. $\nu = \frac{\#\{j; Dv_j - t_j \notin \bar{\Omega}_j\}}{n}$, is smaller when the derogation number is larger. The use of the λ parameter can be understood as follows (this is clear for the algorithm restrained to translations); for $\lambda = 0$ the CR algorithm reduces to the least square method and converges in 1 iteration, but the target is too reduced to get a solution. For $\lambda < 1$, the λ parameter enforces a geometric speed of convergence to the solution by over-relaxing the constraints.

3.4. *Remarks on the implementation of the algorithm*

The first issue we want to address is how to compute the polar decomposition of the matrix H. We chose here to follow the *singular decomposition method* advocated in [4], [7].

The second is the explanation of the convergence test. The algorithm described in section 3 is stopped when $|E(D_{k+1}) - E(D_k)| \leq \epsilon$ for a given real ϵ. In all the numerical tests presented bellow, ϵ is equal to 10^{-6}.

The third one is the following issue: How to chose a good λ parameter? In practise $\lambda < 1$ but close to 1 is advocated in the process (see [13] for a discussion).

4. A Minimizing Derogation Number Algorithm

We have observed in the previous section that the CR algorithm is actually a descent algorithm. The drawback of these algorithms for non convex optimization problems is that the minimizing sequence can be trapped in the neighborhood of a local minimizer. Moreover the search space to minimize the derogation number is huge. Among families of algorithms that have proven their efficiency to handle these problems (local search algorithms, tabu search, simulated annealing, genetic algorithms, ... this list is by no mean exhaustive), stochastic local algorithms are among the most successful and widely used for solving hard combinatorial problems. These algorithms belong to the subclass of Las Vegas algorithms [3]; we refer the interested reader to chapter 4 in [14] for an empirical analysis of performances.

We present below a more precise stochastic local search algorithm to handle the minimization of the derogation number.

4.1. *Stochastic Local Search algorithm*

In mathematics and in computer sciences, stochastic have been used for years to solve optimization problems such as, for example, the Traveling Salesman Problem [15], or even decision problem as satisfiability problem [17, 11, 18, 14, 16]. Below is written the general structure of these algorithms.

Algorithm 1 Local Search algorithm

Require: problem instance π
Ensure: solution $s \in S_\pi$ that is the best candidate solution (with respect to objective function) found at any time during the search

1: $s_m \leftarrow init(\pi)$
2: $s \leftarrow s_m$
3: **while** not $terminate_\pi(s)$ **do**
4: $\quad s' \leftarrow step_\pi(s)$
5: $\quad$ `/* Next test can be included in step function */`
6: $\quad$ **if** $f_\pi(s') < f_\pi(s_m)$ **then**
7: $\quad\quad s_m \leftarrow s'$
8: $\quad$ **end if**
9: $\quad s \leftarrow s'$
10: **end while**
11: **return** s_m

Let us introduce the general framework for our stochastic algorithm (see [12]). Let consider an optimization problem π. Let us define the *search-space* S_π as the

set of possible solutions, that are given as the iterates of a descent algorithm for instance; hence S_π is a discrete space. Let f_π be the fitness function, also called the *objective function*. The basic principle of local search algorithms is to walk among the set S_π, considered as the vertices of a graph, to seek for the solution that achieves the minimum of f_π. To walk along S_π, we need to connect the points of S_π by edges, i.e. to define the notion of neighbors for any given point. To sum up, we then need

- the representation of S_π;
- the *evaluation function* (also called *fitness function*) to evaluate the quality of a candidate solution. This function is often the same as f_π;
- the *neighborhood function* which determines the set of possible solutions s' which are in the neighborhood of each candidate solution s, in order to walk step by step in the search-space;
- the *step function* which determines the next step of the algorithm. This function often uses the evaluation and neighborhood functions. It is very important since it guides the search towards a good quality solution. The difficulty lies in the fact that there might exist some local extrema in the search-space. A local extremum is a candidate solution s such that no neighbor of s has a higher quality than s. One way to deals with local extrema is to introduce some noise in the walk. When probabilities are used to determine the step, this kind of algorithm is called *stochastic*.
- A good initialization, referred as *init*, and a stopping test *terminate* to decide when to stop.

There is no guarantee that the algorithm converges towards the solution which have the best quality. The efficiency of such algorithm strongly depends on its components listed above.

4.2. *A Greedy Minimizing Derogation Algorithm*

In order to minimize derogations, we have developed a *Greedy Minimizing Derogation* (GMD) Algorithm which is a stochastic local search algorithm. It is based on the baselines presented in algorithm 1. Accordingly to the previous subsection, we then introduce (referring also to Section 2.1 for mathematical notations)

- The search-space S is defined as the set of rigid displacements $D = (M, \tau)$. This set is not discrete but in practise we walk along discrete subsequences; actually S is the set a discrete paths given by a descent algorithm such as CR.
- Since the goal of the algorithm is to minimize derogations, we choose for evaluation function the fitness function, namely

$$\begin{aligned} Eval : S &\to \mathbb{N} \\ D &\mapsto N(D) \end{aligned} \tag{4.1}$$

where $N(D)$ is the derogation number defined in (2.2).

- the neighborhood function is defined accordingly to the following rules, that are different respectively to a step using translation or a step using rotations,

$$\begin{aligned} &Neigh : S \to P(S) \\ &\qquad D \mapsto \mathscr{V}_t(D) \cup \mathscr{V}_r(D); \end{aligned} \tag{4.2}$$

where $P(S)$ denotes the set of the parts of S and

▷ The set of the neighborhood of $D = (M, \tau)$ on the translations is defined as

$$\begin{aligned} \mathscr{V}_t(D) = \{ & D' = (M, \tau') \in S : \exists\ \ell \text{ with } Dv_\ell - t_\ell \notin \Omega_\ell \\ & \text{and } \tau' = Dv_\ell - p_\ell(Dv_\ell)\}. \end{aligned} \tag{4.3}$$

▷ The set of the neighborhood of $D = (M, \tau)$ on the rotations is defined as

$$\begin{aligned} \mathscr{V}_r(D) = \{ & D' = (M', \tau) \in S : \exists\ \ell_j\ 1 \le j \le m_1 \text{ with } Dv_{\ell_j} - t_{\ell_j} \notin \Omega_{\ell_j}, \\ & \exists\ \ell_j,\ m_1 + 1 \le j \le m \text{ with } Dv_{\ell_j} - t_{\ell_j} \in \Omega_{\ell_j} \text{ and} \\ & M' \text{ maximizes } I\} \end{aligned} \tag{4.4}$$

where $I(D) = \sum_{j=1}^{m} (Mv_{\ell_j}, p_{\ell_j}(Mv_{\ell_j} - \tau))$. This functional is maximized as in Lemma 3.2.

Remark 4.1. The set $\mathscr{V}_t(D)$ is in bijection with the set of indices $\mathscr{I}_t(D) = \{\ell \in \mathbb{N} : 1 \le \ell \le n,\ Dv_\ell - t_\ell \notin \Omega_\ell\}$, and similarly with the set of indices $\mathscr{I}_r(D)$ with $\mathscr{V}_r(D)$.

- The step function is described in Algorithm 2. Consider a possible solution D and a probability $\bar{\mathscr{P}}$. Chose among the derogated points one point to move inside the constrained domain. This choice is performed either by function *Choose_in_trans* at line 2 or *Choose_in_rot* in line 8; we use more often translations than rotations.

- On the one hand, for *Choose_in_trans*, the idea is to chose the index $\ell \in \mathscr{I}_t(D)$ such that v_ℓ is the nearest point from its Ω_ℓ (this way we lower the perturbation on the other points). Of course, the problem with this method is to produce a ping-pong effect where we go back and forth on a 2-cycle sequence of possible solutions; this drawback is avoided by using some noise, associating to any derogated point the probability to be chosen as

$$\mathscr{P}(\ell) = \frac{1}{|Dv_\ell - t_\ell|} \left(\sum_{v_j | Dv_j - t_j \notin \bar{\Omega}_j} \frac{1}{|Dv_j - t_j|} \right)^{-1}.$$

The chances to chose a point is higher if the point is close to Ω. On the other hand, the function *Choose_in_rot* consists in choosing indices ℓ_j, $1 \le j \le m$ in the set $\mathscr{I}_r(D)$, and maximizing I defined in (4.4) and Lemma 3.2. The discussion on the choice of the parameters m_1 and m are realized in Section 6 below.

Algorithm 2 Step Function

Require: a candidate solution D, a probability $\bar{\mathscr{P}}$, $1 \leq m_1 \leq n$ and $1 \leq m \leq n$
Ensure: the next candidate solution
1: $D' = \text{Choose\_in\_trans}(\mathscr{V}_t(D))$
2: $q_D = Eval(D)$
3: $q_{D'} = Eval(D')$
4: **if** $q_{D'} < q_D$ **then**
5: return D'
6: **else**
7: $D'' = \text{Choose\_in\_rot}(\mathscr{V}_r(D), m_1, m)$
8: $q_{D''} = Eval(D'')$
9: $\mathscr{P}' = \text{random probability in } [0,1]$
10: **if** $q_{D''} < q_D$ **or** $\mathscr{P}' > \bar{\mathscr{P}}$ **then**
11: return D''
12: **else**
13: return D'
14: **end if**
15: **end if**

The global algorithm is then: for any given displacement D, define a neighbor D' using *Choose_in_trans*. If the quality of this neighbor is better than D, it is chosen as next candidate solution; if not new neighbor D' is defined using *Choose_in_rot*. Either the quality of this new point is better and then it is chosen as the next candidate solution, or it is worse and it could be chosen if the probability (line 11) is larger than the threshold value $\bar{\mathscr{P}}$. Here we have introduced some stochasticity through the rotations to avoid local minima.

- The init function is the same as the energy decreasing algorithm initialization described in Section 3.
- The terminate predicate is true when a preset number of step is reached or when the evaluation function reaches 0.

5. CMA-ES: Covariance Matrix Adaptation Evolution Strategy

In this paragraph, we introduce an algorithm, called CMA-ES ([8]), which is an evolutionary algorithm (as genetic algorithm) for difficult non-linear or non-convex optimization problems in continuous domain. Evolutionary algorithms are inspired from biological evolution: reproduction, mutation, recombination and selection. They used theses techniques as operators. They are applied in a loop, called "generation". The generations stop until a terminate criterion is reached.

In an evolution strategy ([5]), new candidate solutions are sampled according to a multivariate normal distribution. Pairwise dependencies between the variables

in this distribution are described by a covariance matrix. The covariance matrix adaptation (CMA) is a method to update the covariance matrix of this distribution.

The sketch of CMA-ES (see [8, 10]) is described in algorithm 3. In this algorithm constants c_c, c_σ, c_{cov}, μ_{cov}, d_σ, μ_{eff}, w_i for $i = 1, \ldots, \mu$ are set to their default values (see [10, 8]). The only parameters are $m \in \mathbb{R}^n$, that is the initial solution and $\sigma \in \mathbb{R}_+$ the step-size.

Algorithm 3 (μ_w, λ)CMA-ES

Require: $m \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$.
Ensure: m the favorite solution
1: $p_\sigma \leftarrow 0$, $p_c \leftarrow 0$, $C \leftarrow I$, $g \leftarrow 0$
2: **while** not terminate **do**
3: •Sample new population of search points
4: **for** $i = 1, \ldots, \mu$ **do**
5: $z_i \sim \mathcal{N}_i(0, C)$
6: $x_i \leftarrow m + \sigma z_i$
7: **end for**
8: •Selection and recombination
9: $\langle z \rangle_{sel} \leftarrow \sum_{i=1}^{\mu} w_i z_{i:\lambda}$ where $\sum_{i=1}^{\mu} w_i = 1$, $w_i > 0$
10: $m \leftarrow m + \sigma \langle z \rangle_{sel}$
11: •Covariance matrix adaptation
12: $p_c \leftarrow (1 - C_c) p_c + \sqrt{1 - (1 - C_c)^2} \sqrt{\mu_{eff}} \langle z \rangle_{sel}$
13: $C \leftarrow (1 - c_{cov}) C + c_{cov} p_c p_c^T + c_{cov} \left(1 - \frac{1}{\mu_{cov}}\right) Z$ where $Z = \sum_{i=1}^{\mu} w_i z_{i:\lambda} z_{i:\lambda}^T$
14: •Step-size control
15: $p_\sigma \leftarrow (1 - C_\sigma) p_\sigma + \sqrt{1 - (1 - C_\sigma)^2} \sqrt{\mu_{eff}}^{\frac{1}{2}} \langle z \rangle_w$
16: $\sigma \leftarrow \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma\|}{E\|\mathcal{N}(0,I)\|} - 1\right)\right)$
17: **end while**
18: **return** (m)

The algorithm starts with sampling a new μ size population of search points (see line 3). This is done by using m, C and σ.

The distribution mean m is updated (see line 8): let $x_{i:\mu}$ be the i-ranked solution point such that $f(x_{1:\mu}) \leq \cdots \leq f(x_{\mu:\mu})$ where f is the fitness function. The best μ parents are selected and weighted intermediate recombination is applied.

Next Covariance matrix C and evolution path p_c are updated (see line 11). The goal of covariance matrix updating is to increase the probability of successful steps $\langle z \rangle_{sel}$ to appear again. Conceptually, the evolution path is the path the strategy takes over a number of generation steps. It can be expressed as a sum of consecutive steps of the mean m.

Finally σ is updated thanks to path length control p_σ (see line 14).

The algorithm iterates until a termination criteria is reached. In general, the algorithm should be stopped whenever it becomes a waste of CPU-time to continue, and it would be better to restart (eventually with increased population size) or to reconsidering the encoding and/or objective function formulation. Many termination criteria have been developed (see for example [2, 9]).

In practice, CMA-ES obtains good results on various problems or benchmarks [6, 9] and on other advantage of CMA-ES is that it takes only two parameters contrary to others same kind algorithms. The drawback of these kind of technique is that there is no guarantee that the minimum is achieved.

6. Numerical results

In this section we provided several numerical results of the different algorithms. All of them have been implemented under Scilab*.

6.1. *Quality criteria for the computed solutions*

In order to evaluate the quality of the computed displacement D, we use several criteria

- The derogation number ($N(D)$).
- The maximal error, that is the distance between the further point to its tolerance domain

$$E_\infty(D) = \max_{j=1\ldots n} |Dv_j - p_j(Dv_j)|.$$

- The least square error, that is the L^2 norm of the tolerance error $E(D)$, defined in (2.4).
- The *free* least square error $Q_2(D)$, without constraints,

$$Q_2(D) = \frac{1}{2}\sum_{j=1}^{n} |Dv_j - t_j|^2.$$

6.2. *First numerical results*

To begin with, we introduce a synthetic example to calibrate the parameters used in our algorithms. Let us describe this example. As seen in Figure 1, we have to align two structures accordingly to the tolerances around 50 points. The tolerance domains Ω_j have the same size and chosen to be a square with a length side equal to 0.05m. Let us emphasize the ratio between the structure and the tolerances (0.2%), and also between the number of degrees of freedom of the system according to the number of constraints (6%). In this artificial example, we know that there exists a displacement D such that $N(D) = E_\infty(D) = E(D) = 0$.

We present now the initial situation, and the results given after the least square iteration.

*http://www.scilab.org/

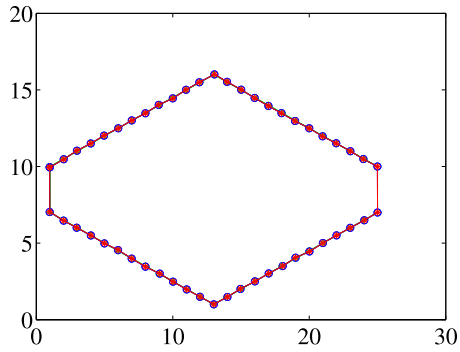

Figure 1. The structure to be assembled in synthetic example

Initial situation ($D_0 = I$):

- $N(D_0) = 3$;
- $E_\infty(D_0) = 1.330 \cdot 10^{-3}$;
- $E(D_0) = 8.393 \cdot 10^{-4}$.
- $Q_2(D_0) = 2.318 \cdot 10^{-1}$.

After the first least square iteration:

- $N(D_1) = 3$;
- $E_\infty(D_1) = 4.454 \cdot 10^{-3}$;
- $E(D_1) = 2.956 \cdot 10^{-3}$.
- $Q_2(D_1) = 2.265 \cdot 10^{-1}$.

We observe that after one iteration three points do not satisfy the constraint requirements.

6.3. *Results of the Constrained/Rigid (CR) Algorithm*

This section is devoted to describe the results obtained by the CR algorithm. After convergence the displacement given by the algorithm is denoted by $\bar{D}$. We discuss here the choice of the parameter λ.

Discussing the λ parameter. In this algorithm, the only parameter is the λ parameter. In Table 1, we can observe the influence of this parameter on the convergence of the method, and on the quality of the computed solutions. The algorithm is performed until $E(D_k) = 0$ or $|E(D_{k+1}) - E(D_k)| \leq \varepsilon$, with $\varepsilon = 10^{-6}$.

We observe in Table 1 that we obtain a solution with a derogation number equal to 0 only for λ equal to 0.95 and 0.98. If $\lambda = 1$, the derogation number is equal to 2, but with points very close from their tolerances ($E(\bar{D}) \simeq E_\infty(\bar{D}) \simeq 3 \times 10^{-5}$). For λ smaller than 0.95, we have only 1 point outside its tolerance, but this point is very far to his tolerance domain, and the convergence is slower. In consequence, the parameter λ will be taken equal to 0.95 in the remaining of this section.

Table 1. Influence of parameter λ, on the synthetic example, for CR algorithm

λ	0.7	0.8	0.9	0.95	0.98	1
Nb of iterations	46	57	91	23	37	132
$N(\bar{D})$	1	1	1	0	0	2
$E_\infty(\bar{D})$	$2.444 \cdot 10^{-3}$	$2.096 \cdot 10^{-3}$	$9.935 \cdot 10^{-5}$	0	0	$3.294 \cdot 10^{-5}$
$E(\bar{D})$	$1.728 \cdot 10^{-3}$	$1.482 \cdot 10^{-3}$	$7.025 \cdot 10^{-5}$	0	0	$2.947 \cdot 10^{-5}$
$Q_2(\bar{D})$	0.229	0.230	0.230	0.228	0.228	0.228

In order to study the behavior of the algorithm during the iterations, we present some plots. Figure 2 shows the number of points outside tolerances versus the iterations ($\lambda = 0.95$). We can observe in this figure that the derogation number is decreasing. This is not always the case and the derogation number could increase among the iterations, for instance see the next example.

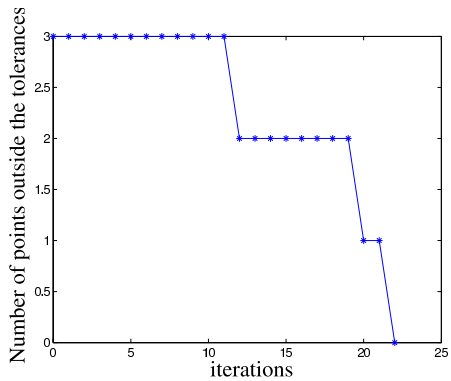

Figure 2. Number of points outside the tolerances versus the iterations (CR algorithm, $\lambda = 0.95$)

Figure 3 shows the evolution of the least square error versus the iterations. The first iteration of the process decreases the free least square error Q_2, while the next iterations increase it. This indicates that the free last square error is not as good as expected as a quality indicator since we have to move points away from the centers of the tolerance domains to decrease the energy E.

Figure 4 shows the evolution of $E(D)$ versus the iterations. Since the very definition of the CR algorithm is a descent algorithm for this energy, we observe that the numerics fit with the theory.

For the sake of completeness, we have also performed a penalty method to solve the minimization process. The penalty method reads: minimize

$$F_\varepsilon(D) = \frac{1}{2}\sum_{i=1}^{n} |Dv_j - t_j|^2 + \frac{1}{\varepsilon}E(D) = Q_2(D) + \frac{1}{\varepsilon}E(D).$$

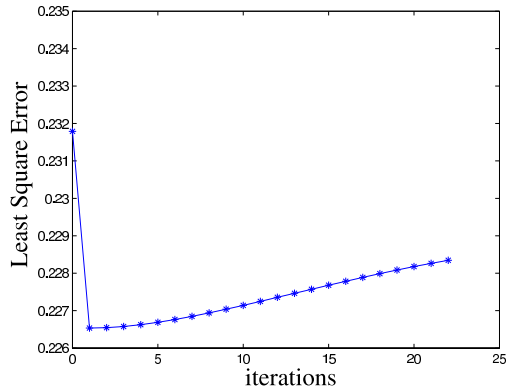

Figure 3. Least square error $Q_2(D_k)$ versus the iterations k (CR algorithm, $\lambda = 0.95$)

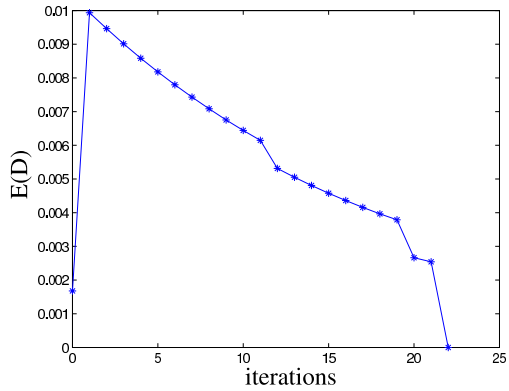

Figure 4. $E(D_k)$ versus the iterations k (CR algorithm, $\lambda = 0.95$)

The drawback of the method is that we have to calibrate another parameter ε. Restricted to translations, this energy is strictly convex and any classical gradient method can be used. For this method, we obtain, for $\varepsilon = 10^{-6}$ and $\lambda = 0.95$, after 2,363 iterations: $N(\bar{D}) = 1$, $E_\infty(\bar{D}) = E(\bar{D}) = 2.76 \cdot 10^{-3}$, $Q_2(\bar{D}) = 0.232$.

6.4. *Results of GMD algorithm*

Since GMD is a stochastic algorithm, then two runs can give two different results. So, in order to evaluate the quality of the results, we provide

- The best result given by the algorithm over all runs.
- The percentage of successful tries defined as the ratio of successful runs over the total number of runs.

- For any quantity as quality indices Q the expectation $\langle Q \rangle$. We run n GMD algorithms and for each run t we compute $Q(t)$ and then we average over the number of runs. For instance if $Q = N(D)$ is the derogation number, $Q(t)$ is the best value reached during the run.
- For any successful run, let us denote by $nsteps$ the number of steps to reach the best solution. We compute then $\langle nsteps \rangle$ that is the average of such quantities over the number of successful runs.
- The worst value possible $\max C$ along a run for a quantity C.

Discussing the parameters m_1 and m. In this paragraph, we pay attention to the influence of the parameters m_1 and m that play a role into the GMD algorithm. Let us recall that m_1 represents the number of derogation points that are very close to the boundary of their tolerance domain while m is the number of insiders that are close to the center of the tolerance domain. For the sake of convenience, and to avoid numerical difficulties, we substitute in the sequel in some places respectively $\min(m, N_{in}(D))$ and $\min(m_1, N(D))$ to of m and m_1, where $N_{in}(D)$ is the number of points inside their tolerance domain.

Due to the numerics in the previous section, we have chosen to use a λ parameter to be either 0.95 or 0.98. Both values have been considered but we only provide the best results obtained with $\lambda = 0.98$ in Table 2. In this simulation, the other parameters are: $Nruns = 100$ and $Nsteps = 50$.

As expected, the rotation step modifies subsequently the system if few points are inside their tolerance domains. While $m = N_{in}(D)$ increases, the rotation step is less efficient to enforce the derogation points to go inside their tolerance domain. Moreover, if m_1 increases one can get a solution that throw points outside their tolerance domain. The chance to get a successful run is bigger for $m_1 = 3$ and $m = 20$; for these values the other quality criteria are successfully satisfied. For these reasons, we fix the values $m_1 = 3$ and $m = 20$ in the experiments below.

Discussing the parameter λ, revisited. We discuss in this paragraph the influence of the parameter λ. For these experiments, the other parameters are set to: $(m_1, m) = (3, 20)$, the number of tries is $Nruns = 100$ and the number of steps for each try is $Nsteps = 50$. Table 3 presents results for λ from 0.7 to 1.

For these experiments, the best percentage of successful tries is obtained for $\lambda = 0.98$. It confirms that it is better to take $\lambda < 1$ but close to 1. We also observe that for λ less than 0.98, the algorithm does not often obtain a solution D such that the derogation number is small (less than 25% of the total number of points). Let us note that for $\lambda = 0.98$ the average number of steps to reach the solution is small, about 7 steps with an initial derogation number $N(D_1) = 3$. So the λ has been set to 0.98 in the experiments to follow.

Table 2. Influence of parameters m and m_1 for GMD algorithm, for the synthetic example ($\lambda = 0.98$, $Nruns = 100$, $Nsteps = 50$)

(m_1, m)	(1,10)	(2,10)	(3,10)	(1,20)	(2,20)	(3,20)
$N(\bar{D})$	0	0	0	0	0	0
% successful tries	13	30	13	92	95	97
$\langle N(D)\rangle$	0.87	0.69	0.87	0.08	0.06	0.01
$\max N(D)$	1	1	1	1	1	1
$\langle E_\infty\rangle(\times 10^{-4})$	1.93	2.17	2.99	0.26	0.13	0.03
$\max E_\infty(\times 10^{-4})$	3.68	3.68	3.68	3.67	3.68	3.68
$\langle E\rangle(\times 10^{-4})$	1.93	2.17	2.99	0.26	0.13	0.03
$\max E(\times 10^{-4})$	3.68	3.68	3.68	3.67	3.68	3.68
$Q_2(\bar{D})$	0.229	0.230	0.229	0.229	0.229	0.229
$\langle Q_2\rangle$	0.230	0.230	0.230	0.229	0.229	0.229
$\max Q_2$	0.231	0.231	0.232	0.230	0.230	0.230

(m_1, m)	$(1,N_{in}(D))$	$(2,N_{in}(D))$	$(3,N_{in}(D))$
$N(\bar{D})$	0	0	0
% successful tries	67	80	83
$\langle N(D)\rangle$	0.32	0.31	0.18
$\max N(D)$	1	1	1
$\langle E_\infty\rangle(\times 10^{-4})$	0.84	0.74	0.35
$\max E_\infty(\times 10^{-4})$	3.68	3.68	3.68
$\langle E\rangle(\times 10^{-4})$	0.84	0.74	0.35
$\max E(\times 10^{-4})$	3.68	3.68	3.68
$Q_2(\bar{D})$	0.231	0.232	0.232
$\langle Q_2\rangle$	0.231	0.229	0.232
$\max Q_2$	0.233	0.232	0.232

Table 3. Influence of parameters λ for GMD algorithm for the synthetic example ($m_1 = 3$, $m = 20$, $Nruns = 100$, $Nsteps = 50$)

λ	0.7	0.8	0.9	0.95	0.98	1
$N(\bar{D})$	0	0	0	0	0	0
% successful tries	6	7	15	23	97	79
$\langle N(D)\rangle$	1.44	1.32	0.84	0.84	0.03	0.21
$\max N(D)$	2	2	2	1	1	1
$\langle Nsteps\rangle_s$	4	17.25	18.5	32.05	6.05	17.46
$\langle E_\infty\rangle$	1.07×10^{-3}	1.27×10^{-4}	8.60×10^{-4}	1.15×10^{-4}	8.07×10^{-6}	1.48×10^{-16}
$\max E_\infty$	4.81×10^{-3}	6.36×10^{-4}	7.69×10^{-3}	1.45×10^{-4}	3.68×10^{-4}	7.07×10^{-16}
$\langle E\rangle$	1.18×10^{-3}	1.20×10^{-4}	8.56×10^{-4}	1.15×10^{-4}	8.07×10^{-6}	1.48×10^{-16}
$\max E$	4.81×10^{-3}	6.37×10^{-4}	7.69×10^{-3}	1.47×10^{-4}	3.68×10^{-4}	7.08×10^{-16}
$Q_2(\bar{D})$	0.230	0.231	0.230	0.232	0.232	0.230
$\langle Q_2\rangle$	0.232	0.232	0.231	0.229	0.229	0.229
$\max Q_2$	0.235	0.237	0.237	0.230	0.232	0.231

6.5. *Results of CMA-ES algorithm*

As we presented it in previous section (see 5), CMAS-ES takes only few parameters: $m \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$. CMA-ES aims to optimize problems in continuous

domain so the functional to minimize is the same as CR algorithm:

$$J(D) = \frac{1}{2}\sum_{j=1}^{n} |Dv_j - p_j(D_k v_j)|^2$$

We ran (3/3_W, 7)-CMA-ES[†] ([8]) where termination criteria are:

- `tolfun`: stop if within-iteration function value differences are smaller than $1e-12$
- `tolfunhist`: stop if function value backward differences are smaller than $1e-12$
- `tolx`: stop if x-changes are smaller than $1e-11 \times \sigma$
- `tolupx`: stop if x-changes are larger than $1e3 \times \sigma$
- `fitness`: target objective function value (minimization)
- `maxfunevals`: maximal number of function evaluations
- `maxiter`: maximal number of iterations (33142)

We have assessed it with different values of m and σ. CMA-ES always found the displacement such that $N(\bar{D}) = 0$ (termination criterion is `fitness`). Let us note that for this example we do not use λ. In Table 4, we present results for $m = [0;0;0]$ and $\sigma = 0.3$. As CMA-ES is an evolutionary algorithm, we provide *Evaluations*: the individual evaluation number (one step generates 7 evaluations). The averaged results $\langle\cdot\rangle$ are computed on 100 runs of CMA-ES.

Table 4. Results obtained with the CMA-ES Algorithm on the synthetic example

$\langle Evaluations\rangle$	504
$N(\bar{D})$	0
$\langle N(D)\rangle$	0
$\langle E_\infty\rangle$	0
$\langle E\rangle$	0
$\langle Q_2\rangle$	0.230
$\max Q_2$	0.233

6.6. *Comparisons*

In this example, all algorithms are able to find solutions with a derogation number equal to zero. Moreover, the solutions have the same quality ($Q_2(\bar{D}) \simeq 0.23$). Only CMA-ES is λ independent.For the other algorithm if the parameter λ is not well chosen (for example $\lambda = 0.8$), only the GMD algorithm is able to find a solution that cancels the derogation number. Observe that in this example, the CR algorithm is less expensive than the greedy GMD algorithm and CMA-ES algorithm. Actually it takes 23 iterations to the CR algorithm. For GMD algorithm, in the worst case for 100 planed runs of 50 steps, it takes $3\% \times 100 \times 50$ (unsuccessful first three

[†]source code can be downloaded on `http://www.lri.fr/~hansen/cmaesintro.html`

tries with 50 steps) +7 (the fourth with an average of 7 steps) = 157 iterations and for CMA-ES it takes 504 evaluations (72 steps).

6.7. *A second example*

The second example has globally the same geometry of the first example (see Figure 1), but the points have been moved a little bit in such a way that $N(D_0)$ is equal to 11 for the initial situation.We first present the quality criteria for the initial situation.
Initial situation ($D_0 = I$):

- $N(D_0) = 11$;
- $E_\infty(D_0) = 3.700 \cdot 10^{-3}$;
- $E(D_0) = 3.470 \cdot 10^{-3}$.
- $Q_2(D_0) = 0.326$;

and after a first least square iteration:

- $N(D_1) = 10$;
- $E_\infty(D_1) = 9.117 \cdot 10^{-3}$;
- $E(D_1) = 1.150 \cdot 10^{-2}$.
- $Q_2(D_1) = 0.322$.

Results given by Constraint/Rigid algorithm, with $\lambda = 0.95$. Results obtained with the CR algorithm, with λ equal to 0.95 are presented in Table 5.

Table 5. Results obtained with Constraint/Rigid algorithm, on the second example

λ	0.95
Nb of iterations	43
$N(\bar{D})$	8
$E_\infty(\bar{D})$	$4.307 \cdot 10^{-3}$
$E(\bar{D})$	$3.684 \cdot 10^{-3}$
$Q_2(\bar{D})$	0.326

After the convergence is achieved (43 iterations), the derogation number is still equal to 8.

Results given by GMD algorithm. In this paragraph we present results obtained with the GMD algorithm. First, Table 6 shows the quality criteria of the best solutions obtained with $Nruns = 100$, $\lambda = 0.98$.

We can observe that with this method, we can obtain displacements $\bar{D}$ such that $N(\bar{D})$ is equal to 4 but the chance to succeed is small (only 3%). It means that for other runs $N(\bar{D})$ is worst than 4. Indeed, the expectation $\langle N(\bar{D}) \rangle$ is almost equal to 5, so the algorithm often converges to 5.

Table 6. Quality and Statistical results obtained with GMD algorithm, on the second example

Nsteps	50
$N(\bar{D})$	4
% successful tries	3
$\langle N(D)\rangle$	5.2
$\max N(D)$	6
$\langle E_\infty\rangle$	$7.263 \cdot 10^{-3}$
$\max E_\infty$	$1.291 \cdot 10^{-2}$
$\langle E\rangle$	$9.691 \cdot 10^{-3}$
$\max E$	$1.456 \cdot 10^{-2}$
$\langle Q_2\rangle$	0.326
$\max Q_2$	0.329
$E_\infty(\bar{D})$	$9.0361 \cdot 10^{-3}$
$E(\bar{D})$	$9.7423 \cdot 10^{-3}$
$Q_2(\bar{D})$	0.327

Discussion on the number of steps for each run. In this paragraph, we study the influence of the number of runs, and the number of steps for each run, in order to optimize the speed of the algorithm and the quality of the best displacement found by GMD algorithm. These experiments have not been made on the synthetic example since the convergence was very fast.

In this simulation, the number of runs is equal to $Nruns = 100$ and the other parameters are $\lambda = 0.98$, $m_1 = 3$ and $m = 20$. The results are provided in Table 7.

Table 7. Influence of the number of steps for GMD algorithm for each run for the second example ($\lambda = 0.98$, $m_1 = 3$, $m = 20$ and $Ntries = 100$)

Nsteps	10	20	30	40	50	60	70
$N(\bar{D})$	4	4	4	4	4	4	4
% successful tries	1	1	2	2	3	3	4
$\langle N(D)\rangle$	6.12	5.71	5.62	5.47	5.20	5.36	5.22
$\max N(D)$	9	8	7	7	6	6	6
$\langle E_\infty\rangle(\times 10^{-3})$	7.00	8.01	7.11	6.88	7.26	7.14	7.19
$\max E_\infty(\times 10^{-2})$	1.35	1.73	1.37	1.23	1.29	1.24	1.39
$\langle E\rangle(\times 10^{-3})$	10.1	10.8	9.73	9.42	9.69	9.69	9.56
$\max E(\times 10^{-2})$	1.81	3.20	1.96	1.61	1.46	1.46	1.39
$Q_2(\bar{D})$	0.327	0.327	0.327	0.327	0.326	0.326	0.326
$\langle Q_2\rangle$	0.326	0.326	0.326	0.326	0.326	0.326	0.326
$\max Q_2$	0.330	0.327	0.331	0.332	0.329	0.328	0.328

These results show that higher the number of steps is, better are $\langle N(D)\rangle$, $\max N(D)$; the number of successful runs also increases with the number of steps

allowed. This is not true for the other criteria. Since increasing the number of steps in each run is costly, we recommend to take less than 50 steps for each run.

Results given by CMA ES algorithm. We ran (3/3_W, 7)-CMA-ES[‡] ([8]) with the same termination criteria as the previous example. We have tried different values for σ and m that have given the same results. In Table 8, we present results for $m = [0;0;0]$ and $\sigma = 0.3$ where termination criteria are `tolfun` and `tolfunhist`. If we introduce λ in this algorithm as other ones, it gives a better $N(\bar{D})$ (the termination criteria are the same that previous) but $\langle E_\infty \rangle$ and $\langle E \rangle$ are greater than the λ independent version as presented in Table 9.

Table 8. Results obtained with CMA-ES Algorithm on the second example

$\langle Evaluations \rangle$	1120
$N(\bar{D})$	11
$\langle N(D) \rangle$	11
$\max N(D)$	11
$\langle E_\infty \rangle$	$3.032 \cdot 10^{-3}$
$\max E_\infty$	$3.032 \cdot 10^{-3}$
$\langle E \rangle$	$3.281 \cdot 10^{-3}$
$\max E$	$3.281 \cdot 10^{-3}$
$\langle Q_2 \rangle$	0.326
$\max Q_2$	0.326

Table 9. Results obtained with CMA-ES Algorithm on the second example with $\lambda = 0.9$

$\langle Evaluations \rangle$	1108
$N(\bar{D})$	6
$\langle N(D) \rangle$	6
$\max N(D)$	6
$\langle E_\infty \rangle$	$5.299 \cdot 10^{-3}$
$\max E_\infty$	$5.299 \cdot 10^{-3}$
$\langle E \rangle$	$4.441 \cdot 10^{-3}$
$\max E$	$4.441 \cdot 10^{-3}$
$\langle Q_2 \rangle$	0.325
$\max Q_2$	0.325

Comparisons. We compare in this paragraph the three algorithms on the second example (Tables 5 to 9). Results show that the CR algorithm is not able to find a displacement $\bar{D}$ such that $N(\bar{D})$ is less than 8 for the second example. Therefore the GMD algorithm can significantly improve the results in this situation; the expectation of the derogation number is smaller and one can achieve $N(D) = 4$. Nevertheless, if the other quality criteria are considered, the results are better for the CR algorithm. In fact, $E(\bar{D})$ is less than 4×10^{-3} for the CR algorithm and more

[‡]Source code can be downloaded on `http://www.lri.fr/~hansen/cmaesintro.html`

than 6×10^{-3} for the GMD algorithm, and $E_\infty(\bar{D})$ is less than 4.4×10^{-3} for the CR algorithm and more than 5×10^{-3} for the GMD algorithm. In other words, each algorithm we provided is better for is own criterion, which is the energy $E(D)$ for the CR algorithm, and the derogation number $N(D)$ for the GMD algorithm. If we compare both algorithm results with CMA-ES ones, without λ parameter it gives better results for $E_\infty(\bar{D})$ and $E(\bar{D})$ than CR one but it is worst for $N(\bar{D})$. With λ parameter sets, conclusions are reversed. In this case, nevertheless it gives better $E_\infty(\bar{D})$ and $E(\bar{D})$ than GDM algorithm but not for $N(\bar{D})$. In this example, one can also observe that each algorithm obtains better results for its own criterium ($E(\bar{D})$ for CR and CMA-ES algorithms, and $N(\bar{D})$ for GMD algorithm).

7. Conclusion

In this article which partakes of mathematical and computer sciences optimization, we have introduced and used one deterministic algorithm, called CR algorithm, that is a descent algorithm for a non convex functional. This algorithm provides us with interesting results. In some particular case, we need a stochastic meta-algorithm, called GMD algorithm, that can decrease significantly the number of derogations numbers in cases where the CR algorithm fails.

References

[1] F. Alouges, A new algorithm for computing liquid crystal stable configurations: the harmonic mapping case, *SIAM Journal on Numerical Analysis* **34**(5) (1997), 1708–1726.

[2] A. Auger and N. Hansen, A restart CMA evolution strategy with increasing population size, in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1769–1776, 2005.

[3] L. Babai, Monte Carlo algorithms in graph isomorphism testing, *Technical Report DMS 79-10*, Universite de Montreal, Montreal, Canada, 1979.

[4] B. Bartoux, Methode "+*n*": Modelisation, *Technical report, LAMFA-CNRS UMR 6140-*, University of Picardie, 2008.

[5] H.-G. Beyer and H.-P. Schwefel, Evolution strategies: a comprehensive introduction, *Journal Natural Computing* **1**(1) (2002), 3–52.

[6] *Special Session on Real-Parameter Optimization of IEEE Congress on Evolutionary Computation* (CEC) 2005, 2005.

[7] G.H. Golub and C.F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, 1996.

[8] N. Hansen, The CMA evolution strategy: a comparing review, in Towards A New Evolutionary Computation. Advances on Estimation of Distribution Algorithms, pp. 75–102, Springer, 2006.

[9] N. Hansen, Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed, in *Workshop Proceedings of the GECCO Genetic and Evolutionary Computation Conference*, pp. 2389–2395, 2009.

[10] N. Hansens, *The CMA Evolution Strategy: A Tutorial*, Tutorial, March 2010.

[11] E.A. Hirsch and A. Kojevnikov, Unitwalk: A new SAT solver that uses local search guided by unit clause elimination, *Annals of Mathematics and Artificial Intelligence* **43**(1-4) (2005), 91–111.

[12] H.H. Hoos and T. Stutzle, *STOCHASTIC LOCAL SEARCH Foundations and Applications*, Elsevier, 2005.

[13] S. Lefebvre, Modeles mathematiques et outils informatiques pour lĄfassemblage de structures aeronotiques par balancement spacial, *Ph.D. Thesis, LAMFA-CNRS UMR 6140-*, University of Picardie, 2009.

[14] C.M. Li, W. Wei and H. Zhang, Combining adaptive noise and look-ahead in local search for SAT, in *Proceedings of 10th International Conference on the Theory and Applications of Satisfiability Testing* (SAT2007), pp. 121–133, Lisbon, Portugal, May 2007.

[15] S. Lin, Computer solutions of the traveling salesman problem, *Bell System Technical Journal* **44**(1965), 2245–2269.

[16] D. Pham, J. Thornton, C. Gretton and A. Sattar, Combining adaptive and dynamic local search for satisfiability, *Journal on Satisfiability, Boolean Model Checking and Computation*, 2008.

[17] B. Selman, H.J. Levesque and D. Mitchell, Gsat: a new method for solving hard satisfiability problems, in *Proceedings of the Tenth National Conference on Artificial Intelligence* (AAAI'92), pp. 440–446, 1992.

[18] D.A.D. Tompkins and H.H. Hoos, On the quality and quantity of random decisions in stochastic local search for SAT, in *Proceedings of the 19th Conference of the Canadian Society for Computational Studies of Intelligence* (AI-2006), *Vol. 4013 of Lecture Notes in Artificial Intelligence*, pp. 146–158, 2006.

Laure Devendeville, *MIS EA 4290, Université de Picardie Jules Verne, 33 rue Saint-Leu, 80039 Amiens cedex 1, France*.
E-mail: laure.devendeville@u-picardie.fr

Serge Dumont, *LAMFA CNRS UMR 7352, Université de Picardie Jules Verne, 33 rue Saint-Leu, 80039 Amiens cedex 1, France*.
E-mail: serge.dumont@u-picardie.fr

Olivier Gubet, *LAMFA CNRS UMR 7352, Université de Picardie Jules Verne, 33 rue Saint-Leu, 80039 Amiens cedex 1, France*.
E-mail: olivier.goubet@u-picardie.fr

Sylvain Lefebvre, *LAMFA CNRS UMR 7352, Université de Picardie Jules Verne, 33 rue Saint-Leu, 80039 Amiens cedex 1, France*.
E-mail: sylvain.lefebvre@u-picardie.fr

Received January 21, 2012
Accepted May 23, 2012