Conjugate Gradient Algorithms for Molecular Formation under pairwise Potential Minimization

Neculai Andrei

The paper presents a survey of nonlinear conjugate gradient algorithms and the computational performances of some conjugate gradient algorithms for solving the molecular conformation application, i.e. the so called the Lennard-Jones clusters problem. The conjugate gradient algorithms classifies in four groups: conjugacy condition algorithms, hybrid conjugate gradient, scaled conjugate gradient and preconditioned conjugate gradient algorithms. In this paper we present the main characteristics of these conjugate gradient algorithms emphasizing their convergence properties. We present the performances of 25 conjugate gradient algorithms to solve the protein folding problem. We show that the conjugate gradient algorithms are able to solve this difficult problem, and our scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization is the top performer among the conjugate gradient algorithms.

Key words. Molecular formation, potential minimization, nonlinear conjugate gradient methods, unconstrained optimization, nonlinear programming.

1. Introduction

One of the very important problem with a crucial role in structure determination and analysis of proteins, peptides and other organic molecules is energy minimization. The interatomic interactions of biomolecules such as proteins and nucleic acids are described by an empirical potential energy function (force field) which depends on the structure, i.e. on the geometry of the molecule, and typically leads to an energy surface with a tremendous number of local minima. Identifying the lowest energy minima is the goal of protein folding problem. This problem received a great deal of attention both from its importance in computational structural biology in analysis.

* Research Institute for Informatics, Center for Advanced Modeling and Optimization 8-10, Averescu Avenue, Bucharest 1, Romania, e-mail: nandrei@ici.ro
of organic aggregates and in testing the performances of unconstrained optimization algorithms.

The algorithmic and computational studies given by Navon et al. [1992a, 1992b], Wang, et al. [1995] and Zou et al. [1993] show that truncated Newton (TN, [Nash, 1984]) and limited BFGS (L-BFGS, [Liu and Nocedal, 1989]) are powerful optimization methods that are more efficient than other techniques. Truncated Newton method combine the quadratic convergence rate of the classic Newton method with feasible storage and computational requirements. On the other hand, the L-BFGS algorithm is simple to use mainly because it does not require the knowledge of the sparsity structure of the Hessian or knowledge of the separability of the minimization function. The conclusion is that the TN performs better than L-BFGS for functions that are nearly quadratic, while for highly nonlinear complex functions L-BFGS outperforms TN.

An excellent review on minimization methods has been considered by Schlick [1992] where some details on TNPACK package co-authored with Fogelson, which implements a TN algorithm, are presented [Schlick and Fogelson, 1992]. The main characteristic of TNPACK is that the user has the possibility to supply a sparse preconditioning matrix that cluster the eigenvalues of the preconditioned Hessian, thus accelerating the convergence of the optimization process. The second implementation of TN algorithm is that given by Nash [1984] which uses an automatic preconditioning. The Nash’s implementation TN has the advantage of portability because the preconditioner does not have to be tailored to a specific problem. Some computational studies and comparisons between several algorithms applied to the molecule deoxycytine and to cluster of water molecules are presented by Schlick. It is shown that for molecule deoxyctyne the TN with preconditioning is the most efficient requiring about half of the CPU time corresponding to L-BFGS with preconditioning.

In this paper we present another approach to models of protein by using the conjugate gradient algorithms for unconstrained optimization. Conjugate gradient methods represent a class of unconstrained optimization algorithms characterized by low memory requirements and strong local and global convergence properties. The structure of the paper is as follows. In section 2 we present a survey of nonlinear conjugate gradient algorithms. Section 3 is dedicated to present the Lennard-Jones Clusters, i.e. the molecular conformation problem and its solution given by 25 nonlinear conjugate gradient algorithms. It is shown that our scaled memoryless BFGS preconditioned conjugate gradient algorithm is the top performer among the conjugate gradient algorithms [Andrei, 2006a]. Some conclusions are given in section 4.

2. A survey of nonlinear conjugate gradient algorithms

An excellent survey of nonlinear conjugate gradient methods, with special attention to global convergence properties, has been given by Hager and Zhang [2005]. In this section we focus on different versions of nonlinear conjugate gradient methods and their convergence properties. Consider the problem to minimize a function of $n$ variables:
Conjugate gradient algorithms

\[ \min f(x), \quad (1) \]

where \( f : \mathbb{R}^n \to \mathbb{R} \) is a continuous differentiable function with \( g(x) = \nabla f(x) \) its gradient. Conjugate gradient algorithms solving (1) are iterative methods of the form:

\[ x_{k+1} = x_k + \alpha_k d_k, \quad (2) \]

where \( \alpha_k > 0 \) is a step length and \( d_k \) is a search direction. The search direction at the very first iteration is the steepest descent direction:

\[ d_0 = -g_0 = \nabla f(x_0). \]

The directions along the iterations are computed according to:

\[ d_{k+1} = -g_{k+1} + \beta_k d_k, \quad (3) \]

where \( \beta_k \) is a scalar. The nonlinear conjugate gradient algorithm for general unconstrained optimization problem was firstly proposed by Fletcher and Reeves [1964]. In this case the conjugacy condition is replaced by:

\[ d_k^T y_k = 0, \]

where \( y_k = g_{k+1} - g_k \). This is motivated by the following relations:

\[ d_k^T A y_k = \frac{1}{\alpha_k} d_k^T A (x_{k+1} - x_k) = \frac{1}{\alpha_k} d_k^T (g_{k+1} - g_k) = \frac{1}{\alpha_k} d_k^T y_k, \]

or by the mean value theorem

\[ d_k^T y_k = \alpha_k \beta \nabla^2 f(x + \theta \alpha_k d_k) d_k, \]

for some \( \theta \in (0, 1) \).

In order to ensure the convergence of algorithm (2), it is necessary to constrain the choice of \( \alpha_k \). Usually, the step length is selected to satisfy the Wolfe line search conditions [Wolfe, 1969, 1971]:

\[ f(x_k + \alpha_k d_k) - f(x_k) \leq \sigma_1 \alpha_k g_k^T d_k, \quad (4) \]

\[ \nabla f(x_k + \alpha_k d_k)^T d_k \geq \sigma_2 g_k^T d_k, \quad (5) \]

where \( 0 < \sigma_1 \leq \sigma_2 < 1 \). For some conjugate gradient algorithms the stronger versions of the Wolfe line search conditions are needed to ensure the convergence and the stability. The strong Wolfe conditions consists of (4) and the following strengthened version of (5)

\[ |\nabla f(x_k + \alpha_k d_k)^T d_k| \leq \sigma_2 g_k^T d_k. \quad (6) \]
The general conjugate gradient algorithm

Step 1. Select \( x_0 \), set \( d_0 = -g_0 \) and \( k = 0 \).

Step 2. Compute the step length \( \alpha_k > 0 \) satisfying the Wolfe line search (4) and (5).

Step 3. Compute \( x_{k+1} = x_k + \alpha_k d_k \). If \( \|g_{k+1}\| \leq \varepsilon \), then stop.

Step 4. Compute \( \beta_k \) and generate the direction \( d_{k+1} = -g_{k+1} + \beta_k d_k \).

Step 5. Set \( k = k + 1 \) and go to step 2.

2.1. Classical Conjugate Gradient Algorithms

Different conjugate gradient algorithms correspond to different choices for the scalar \( \beta_k \). The well known formulas for \( \beta_k \) are summarized in Table 1. Conjugate gradient algorithms (2)–(3) with exact line searches satisfy the equality

\[
g_k^T d_k = -\|g_k\|^2,
\]

which implies the sufficient descent condition

\[
g_k^T d_k < -c\|g_k\|^2,
\]

where \( c > 0 \) is a constant. Often, the sufficient descent condition has been used to analyze the global convergence of conjugate gradient algorithms with inexact line searches.

2.2. Hybrid and parametrized conjugate gradient algorithms

The numerical experiments show that although the FR, DY and CD conjugate gradients methods have strong convergence properties; however they may be affected by jamming. On the other hand, the HS, PRP and LS methods although theoretically may not converge, computationally they often are significantly better. The idea to combine these methods in order to get efficient algorithms leads to hybrid conjugate gradient algorithms. Table 2 contains the main hybrid conjugate gradient methods.

Conjugate gradient methods can be combined to get the so called parameter conjugate gradient algorithms. Dai and Yuan [1998, 2003] proposed a one-parameter family of conjugate gradient algorithms with:

\[
\beta_k = \frac{\|g_{k+1}\|^2}{\lambda_k \|g_k\|^2 + (1 - \lambda_k)d_k^T y_k},
\]

where \( \lambda_k \in [0, 1] \) is a parameter. For \( \lambda_k = 1 \) we get the Fletcher-Reeves method, while the Dai-Yuan method correspond to \( \lambda_k = 0 \). By considering convex combinations of the numerators and denominators of \( \beta_k^{FR} \) and \( \beta_k^{HS} \), Nazareth [1999] suggested a two-parameter family of conjugate gradient methods:

\[
\beta_k = \frac{\mu_k \|g_{k+1}\|^2}{\lambda_k \|g_k\|^2 + (1 - \lambda_k)\|d_k\|^2 y_k},
\]

where \( \lambda_k, \mu_k \in [0, 1] \) are parameters.
Table 1. Choices for $\beta_k$ parameter in conjugate gradient

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta_{HS}$</td>
<td>$\frac{g_{k+1}^Ty_k}{d_k^Ty_k}$</td>
</tr>
<tr>
<td>$\beta_{FR}$</td>
<td>$\frac{g_{k+1}^Tg_k + g_k^Ty_k}{g_k^Ty_k}$</td>
</tr>
<tr>
<td>$\beta_D$</td>
<td>$\frac{g_{k+1}^T\nabla^2 f(x_k)d_k}{d_k^T\nabla^2 f(x_k)d_k}$</td>
</tr>
<tr>
<td>$\beta_{PRP}$</td>
<td>$\frac{g_{k+1}^Ty_k}{g_k^Ty_k}$</td>
</tr>
<tr>
<td>$\beta_{PRP+}$</td>
<td>$\max\left{0, \frac{g_{k+1}^Ty_k}{g_k^Ty_k}\right}$</td>
</tr>
<tr>
<td>$\beta_{CD}$</td>
<td>$\frac{g_{k+1}^Tg_k + g_k^Ty_k}{d_k^T\nabla^2 f(x_k)d_k}$</td>
</tr>
<tr>
<td>$\beta_{LS}$</td>
<td>$\frac{g_{k+1}^Tg_k + g_k^Ty_k}{d_k^T\nabla^2 f(x_k)d_k}$</td>
</tr>
<tr>
<td>$\beta_{DY}$</td>
<td>$\frac{g_{k+1}^Tg_k + g_k^Ty_k}{d_k^T\nabla^2 f(x_k)d_k}$</td>
</tr>
<tr>
<td>$\beta_{DL}^+$</td>
<td>$\max\left{0, \frac{g_{k+1}^Tg_k + g_k^Ty_k}{d_k^T\nabla^2 f(x_k)d_k}\right}$, $t &gt; 0$</td>
</tr>
<tr>
<td>$\beta_{YT}$</td>
<td>$\frac{g_{k+1}^Tg_k + g_k^Ty_k}{d_k^T\nabla^2 f(x_k)d_k}$</td>
</tr>
<tr>
<td>$\beta_{YT+}$</td>
<td>$\max\left{0, \frac{g_{k+1}^Tg_k + g_k^Ty_k}{d_k^T\nabla^2 f(x_k)d_k}\right}$, $t &gt; 0$</td>
</tr>
</tbody>
</table>

where $\lambda_k$, $\mu_k \in [0, 1]$. Observe that this two-parameter family includes FR, DY, PRP and HS methods. Dai and Yuan [2001] considered a three-parameter family of hybrid conjugate gradient method; they chose:

$$
\beta_k = \frac{\mu_k \|g_{k+1}\|^2 + (1 - \mu_k)g_{k+1}^Ty_k}{(1 - \lambda_k - \omega_k) \|g_k\|^2 + \lambda_k d_k^Ty_k - \omega_k d_k^Ty_k},
$$

(11)

where $\lambda_k$, $\mu_k \in [0, 1]$ and $\omega_k \in [0, 1 - \lambda_k]$. This three-parameter family includes the six standard conjugate gradient methods, the previous one-parameter and two-parameter families, as well as many hybrid methods as special cases.

2.3. Scaled Conjugate Gradient Algorithms

Another class of conjugate gradient algorithms is given by the so called the scaled conjugate gradient algorithms. For these algorithms the direction is computed
Table 2. Hybrid choices for the $\beta_k$ parameter in conjugate gradient algorithms

<table>
<thead>
<tr>
<th>$\beta_k^{TAS}$</th>
<th>Proposed by Touat-Ahmed and Storey [1990].</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta_k^{P RP}$, $0 \leq \beta_k^{P RP} \leq \beta_k^{F R}$, $\beta_k^{F R}$, otherwise.</td>
<td></td>
</tr>
<tr>
<td>$\beta_k^{HSt} = \max {0, \min {\beta_k^{P RP}, \beta_k^{F R}}}$</td>
<td>Proposed by Hu and Storey [1991].</td>
</tr>
<tr>
<td>$\beta_k^{FR} max {-\beta_k^{F R}, \min {\beta_k^{P RP}, \beta_k^{F R}}}$</td>
<td>Proposed by Gilbert and Nocedal [1992].</td>
</tr>
<tr>
<td>$\beta_k^{DY} = \max {0, \min {\beta_k^{HSt}, \beta_k^{DY}}}$</td>
<td>Proposed by Dai and Yuan [2001] and Dai and Ni [2003].</td>
</tr>
<tr>
<td>$\beta_k^{DY} = \max {-\beta_k^{DY}, \min {\beta_k^{HSt}, \beta_k^{DY}}}$</td>
<td>Proposed by Dai and Yuan [2001]. $\sigma$ is the parameter used in the second Wolfe line search condition.</td>
</tr>
<tr>
<td>$\beta_k^{LS-CD} = \max {0, \min {\beta_k^{L S}, \beta_k^{C D}}}$</td>
<td>Proposed by Dai and Yuan [2001]. $\sigma$ is the parameter used in the second Wolfe line search condition.</td>
</tr>
</tbody>
</table>

as:

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k d_k,$$

where $\theta_{k+1}$ is a positive parameter (the scaling parameter).

Observe that if $\theta_{k+1} = 1$, then we get the classical conjugate gradient algorithms according to the value of the scalar parameter $\beta_k$. On the other hand, if $\beta_k = 0$, then we get another class of algorithms according to the selection of the parameter $\theta_{k+1}$. There are two possibilities for $\theta_{k+1}$: a positive scalar or a positive definite matrix. If $\theta_{k+1} = 1$, we have the steepest descent algorithm. If $\theta_{k+1} = \nabla^2 f(x_{k+1})^{-1}$, or an approximation of it, then we get the Newton or the quasi-Newton algorithms, respectively. Therefore, we see that in the general case, when $\theta_{k+1} \neq 0$ is selected in a quasi-Newton manner and $\beta_k \neq 0$, then (12) represents a combination between the quasi-Newton and conjugate gradient methods.

To determine $\beta_k$, Andrei [2004a, 2004b, 2005, 2006a, 2006b] considers the following procedure. As we know the Newton direction for solving (1) is given by $d_{k+1} = -\nabla^2 f(x_{k+1})^{-1} g_{k+1}$. Therefore, from the equality

$$-\nabla^2 f(x_{k+1})^{-1} g_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k d_k,$$

we get:

$$\beta_k = \frac{s_k^T \nabla^2 f(x_{k+1}) \theta_{k+1} g_{k+1} - s_k^T g_{k+1}}{s_k^T \nabla^2 f(x_{k+1}) s_k}.\quad (13)$$

Using the Taylor development, after some algebra we obtain:
\[
\beta_k = \frac{(\theta_{k+1}y_k - s_k)^T g_{k+1}}{y_k^T s_k}, \tag{14}
\]

where \(s_k = x_{k+1} - x_k\) and \(y_k = g_{k+1} - g_k\).

If \(\theta_{k+1} = 1\), then (14) is the parameter corresponding to the direction considered by Perry [1977], i.e. we get the scaled Perry algorithm. This value of parameter \(\beta_k\) is used by Birgin and Martínez [2001] in their SCG (spectral conjugate gradient) package for unconstrained optimization, where \(\theta_{k+1}\) is selected in a spectral manner, as suggested by Raydan [1997]. The following particularizations can be remarked. If \(s_j^T g_{j+1} = 0, j = 0, 1, \ldots, k\), then we get a generalization of the Polak and Ribiére formula [1969], i.e. the scaled Polak and Ribiére formula. If \(s_j^T g_{j+1} = 0, j = 0, 1, \ldots, k\), and additionally the successive gradients are orthogonal, then we obtain a generalization of the Fletcher and Reeves formula [1964], i.e. the scaled Fletcher and Reeves formula. Table 3 summarizes the scaled choices for the parameter \(\beta_k\).

<table>
<thead>
<tr>
<th>(\beta_k)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\beta_k^{SP} = \frac{\theta_{k+1}y_k - s_k}{y_k^T s_k})</td>
<td>Scaled Perry. Suggested by Birgin and Martínez [2001] and Andrei [2004a, 2004b, 2005, 2006a, 2006b].</td>
</tr>
<tr>
<td>(\beta_k^{SP+} = \max \left{ 0, \frac{\theta_{k+1}y_k - s_k}{y_k^T s_k} \right} - \frac{\theta_{k+1}y_k}{y_k^T s_k})</td>
<td>Scaled Perry +. Suggested by Birgin and Martínez [2001].</td>
</tr>
<tr>
<td>(\beta_k^{SPRP} = \frac{\theta_{k+1}y_k}{\alpha_k y_k - g_k^T g_k})</td>
<td>Scaled Polak-Ribiére-Polyak. Suggested by Birgin and Martínez [2001] and Andrei [2004a, 2004b, 2005, 2006a, 2006b].</td>
</tr>
<tr>
<td>(\beta_k^{SPR} = \frac{\theta_{k+1}y_k}{\alpha_k y_k - g_k^T g_k})</td>
<td>Scaled Fletcher-Reeves. Suggested by Birgin and Martínez [2001] and Andrei [2004a, 2004b, 2005, 2006a, 2006b].</td>
</tr>
</tbody>
</table>

The parameter \(\theta_{k+1}\) is selected in a spectral manner, \(\theta^s\), as the inverse of the Rayleigh quotient or in an anticipative manner, \(\theta^a\), as given in Andrei [2004a, 2004b, 2005, 2006a, 2006b].

### 2.4. BFGS preconditioned conjugate gradient algorithm (SCALCG Algorithm)

There is a result of Shanno [1978] that says that the conjugate gradient method is the BFGS quasi-Newton method for which, at every iteration, the initial approximation to the inverse of the Hessian is taken as the identity matrix. Shanno [1978] shows how the traditional Fletcher-Reeves and Polak-Ribiére conjugate gradient algorithms may be modified in a form established by Perry to a sequence which can be considered as a memoryless BFGS preconditioned. The algorithm is embedded into a restarting procedure based on Powell’s restart criterion. The idea of the algorithm is to modify the direction in such a manner to overcome the lack of positive definiteness.
of the matrix defining the search direction. The CONMIN algorithm based on this technique proved to be one of the most powerful for solving large-scale unconstrained optimization problems. The extension of the preconditioning technique to the scaled conjugate gradient is very simple. Using the same methodology Andrei [2004a, 2004b, 2005, 2006a, 2006b] obtained the following direction $d_{k+1}$:

$$
d_{k+1} = -\theta_{k+1}g_{k+1} + \theta_{k+1} \left( \frac{y_k^T s_k}{y_k^T s_k} \right) y_k - \left[ 1 + \theta_{k+1} \frac{y_k^T y_k}{y_k^T s_k} \right] \frac{y_k^T s_k}{y_k^T s_k} - \theta_{k+1} \frac{y_k^T y_k}{y_k^T s_k} s_k,
$$

Thus, we get another class of scaled memoryless BFGS preconditioned conjugate gradient algorithms. The implementation of the direction (15), in the context of Powell restarts, is given by Andrei [2006a, 2006b] in SCALCG package. In SCALCG algorithm $\theta_{k+1}$ is defined as a scalar approximation of the inverse Hessian. The following procedures can be used: $\theta_{k+1}$ spectral or $\theta_{k+1}$ anticipative [Andrei 2004a].

---

**Algorithm SCALCG**

**Step 1.** Select $x_0 \in \mathbb{R}^n$, and the parameters $0 < \sigma_1 \leq \sigma_2 < 1$. Compute $f(x_0)$ and $g_0 = \nabla f(x_0)$. Set $d_0 = -g_0$ and $\alpha_0 = 1/\|g_0\|$. Set $k = 0$.

**Step 2.** Line search. Compute $\alpha_k$ satisfying the Wolfe conditions (4) and (5). Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1})$, $g_{k+1}$ and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

**Step 3.** Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set $k = k + 1$.

**Step 4.** Compute $\theta_k$ using a spectral $\theta_{k+1} = s_k^T s_k/y_k^T s_k$ or an anticipative $\theta_{k+1} = 1/\gamma_{k+1}$ approach, where $\gamma_{k+1} = \frac{2 \alpha_k^2}{\alpha_k(1-\alpha_k)} [f(x_{k+1}) - f(x_k) - \alpha_k g_k^T d_k]$.

**Step 5.** Compute the (restart) direction $d_k$ as in (15).

**Step 6.** Line search. Compute the initial guess of the step length as $\alpha_k = \alpha_{k-1} \|d_{k-1}\|_2 / \|d_k\|_2$. Using this initialization compute $\alpha_k$ satisfying the Wolfe conditions (4) and (5). Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1})$, $g_{k+1}$ and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

**Step 7.** Store $\theta = \theta_k$, $s = s_k$ and $y = y_k$. 

---
where the stepsize $\alpha$ is continuously differentiable, recently Andrei [2006c] considered a nonlinear conjugate gradient algorithm:

\[ \text{Conjugate gradient algorithms 15} \]

\[ \text{Step 8.} \quad \text{Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set } k = k + 1. \]

\[ \text{Step 9.} \quad \text{If the Powell restart criterion } |g_{r+1}^T g_r| \geq 0.2 \|g_{r+1}\|^2, \text{ or the angle restart criterion } d_r^T g_{r+1} > -10^{-3} \|d_r\| \|g_{r+1}\|, \text{ is satisfied, then go to step 4 (a restart step); otherwise continue with step 10 (a standard step).} \]

\[ \text{Step 10. Compute } \]

\[ v = \theta y_k - \theta \left( \frac{g_k^T s_k}{s_k^T s_k} \right) y \]

\[ w = \theta y_k - \theta \left( \frac{g_k^T s_k}{s_k^T s_k} \right) y + \left( 1 + \theta \frac{y_k^T y_k}{y_k^T s_k} \right) \frac{g_k^T s_k}{s_k^T s_k} y \]

\[ d_k = -v + \frac{(g_k^T s_k) w + (g_k^T w) s_k}{y_k^T y_k} \frac{g_k^T s_k}{y_k^T s_k} s_{k-1}. \]

\[ \text{Step 11.} \quad \text{Line search. Compute the initial guess of the step length as } \alpha_k = \alpha_{k-1} \|d_{k-1}\| / \|d_k\|. \] Using this initialization compute $\alpha_k$ satisfying the Wolfe conditions (4) and (5). Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1})$, $g_{k+1}$ and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

\[ \text{Step 12.} \quad \text{Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set } k = k + 1 \text{ and go to step 9.} \]

It is well known that if $f$ is bounded below along the direction $d_k$, then there exists a step length $\alpha_k$ satisfying the Wolfe conditions. The initial selection of the step length crucially affects the practical behavior of the algorithm. At every iteration $k \geq 1$ the starting guess for the step $\alpha_k$ in the line search is computed as $\alpha_{k-1} \|d_{k-1}\| / \|d_k\|$. This selection, was considered for the first time by Shanno and Phua in CONMIN [1976].

### 2.5. Conjugate Gradient Algorithms with Sufficient Descent Condition

For solving the unconstrained optimization problem (1) where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is continuously differentiable, recently Andrei [2006c] consider a nonlinear conjugate gradient algorithm:

\[ x_{k+1} = x_k + \alpha_k d_k, \]

where the stepsizes $\alpha_k$ is positive and the directions $d_k$ are computed by the rule:

\[ d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_{k} s_k, \quad d_0 = -g_0, \]

\[ \theta_{k+1} = \frac{g_k^T s_k}{g_{k+1}^T s_{k+1}}, \]

\[ \text{Step 8.} \quad \text{Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set } k = k + 1. \]

\[ \text{Step 9.} \quad \text{If the Powell restart criterion } |g_{r+1}^T g_r| \geq 0.2 \|g_{r+1}\|^2, \text{ or the angle restart criterion } d_r^T g_{r+1} > -10^{-3} \|d_r\| \|g_{r+1}\|, \text{ is satisfied, then go to step 4 (a restart step); otherwise continue with step 10 (a standard step).} \]

\[ \text{Step 10. Compute } \]

\[ v = \theta y_k - \theta \left( \frac{g_k^T s_k}{s_k^T s_k} \right) y \]

\[ w = \theta y_k - \theta \left( \frac{g_k^T s_k}{s_k^T s_k} \right) y + \left( 1 + \theta \frac{y_k^T y_k}{y_k^T s_k} \right) \frac{g_k^T s_k}{s_k^T s_k} y \]

\[ d_k = -v + \frac{(g_k^T s_k) w + (g_k^T w) s_k}{y_k^T y_k} \frac{g_k^T s_k}{y_k^T s_k} s_{k-1}. \]

\[ \text{Step 11.} \quad \text{Line search. Compute the initial guess of the step length as } \alpha_k = \alpha_{k-1} \|d_{k-1}\| / \|d_k\|. \] Using this initialization compute $\alpha_k$ satisfying the Wolfe conditions (4) and (5). Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1})$, $g_{k+1}$ and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

\[ \text{Step 12.} \quad \text{Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set } k = k + 1 \text{ and go to step 9.} \]

It is well known that if $f$ is bounded below along the direction $d_k$, then there exists a step length $\alpha_k$ satisfying the Wolfe conditions. The initial selection of the step length crucially affects the practical behavior of the algorithm. At every iteration $k \geq 1$ the starting guess for the step $\alpha_k$ in the line search is computed as $\alpha_{k-1} \|d_{k-1}\| / \|d_k\|$. This selection, was considered for the first time by Shanno and Phua in CONMIN [1976].

### 2.5. Conjugate Gradient Algorithms with Sufficient Descent Condition

For solving the unconstrained optimization problem (1) where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is continuously differentiable, recently Andrei [2006c] consider a nonlinear conjugate gradient algorithm:

\[ x_{k+1} = x_k + \alpha_k d_k, \]

where the stepsizes $\alpha_k$ is positive and the directions $d_k$ are computed by the rule:

\[ d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_{k} s_k, \quad d_0 = -g_0, \]

\[ \theta_{k+1} = \frac{g_k^T s_k}{g_{k+1}^T s_{k+1}}, \]
\[ \beta_k^{CGSD} = \frac{1}{y_k^T s_k} \left( g_{k+1} - \delta_k \frac{\|g_{k+1}\|^2}{y_k^T s_k} s_k \right)^T g_{k+1}, \]  
\[ \delta_k = \frac{y_k^T g_{k+1}}{g_{k+1}^T g_{k+1}}. \]  

Here \( g_k = \nabla f(x_k) \) and \( y_k = g_{k+1} - g_k, \) \( s_k = x_{k+1} - x_k. \) Observe that if \( f \) is a quadratic function and \( \alpha_k \) is selected to achieve the exact minimum of \( f \) in the direction \( d_k, \) then \( s_k^T g_{k+1} = 0 \) and the formula (20) for \( \beta_k^{CGSD} \) reduces to the Dai and Yuan computational scheme. However, in this paper we consider general nonlinear functions and inexact line search.

This computational scheme has been obtained by modifying the Dai and Yuan scheme \( \beta_k^{DY} = g_{k+1}^T g_{k+1}/y_k^T s_k, \) in order to satisfy both the sufficient descent and conjugacy conditions in the frame of conjugate gradient methods. To determine the parameters \( \theta_{k+1} \) and \( \delta_k, \) from (18) and (20), after some algebra we get:

\[ d_{k+1} = -Q_{k+1} g_{k+1}, \]  

where

\[ Q_{k+1} = \theta_{k+1} I - \frac{s_k g_{k+1}^T}{y_k^T s_k} + \delta_k \frac{\|g_{k+1}\|^2}{(y_k^T s_k)^2} (s_k s_k^T). \]  

Now, by symmetrization of \( Q_{k+1} \) as:

\[ \bar{Q}_{k+1} = \theta_{k+1} I - \frac{s_k g_{k+1}^T}{y_k^T s_k} + \delta_k \frac{\|g_{k+1}\|^2}{(y_k^T s_k)^2} (s_k s_k^T), \]  

we can consider \( d_{k+1} = -\bar{Q}_{k+1} g_{k+1}. \) Therefore, from the conjugacy condition, \( y_k^T d_{k+1} = 0, \) i.e.

\[ y_k^T \bar{Q}_{k+1} = 0, \]

it follows that:

\[ \theta_{k+1} = \frac{g_{k+1}^T g_{k+1}}{y_k^T g_{k+1}} \quad \text{and} \quad \delta_k = \frac{y_k^T g_{k+1}}{y_k^T g_{k+1}} = \frac{1}{\bar{\delta}_{k+1}}. \]  

To conclude the sufficient descent condition from (22), the quantity \( \theta_{k+1} - 1/(4\delta_k) \) is required to be nonnegative. Supposing that \( \theta_{k+1} - 1/(4\delta_k) > 0, \) then the direction given by (18) and (20) is a descent direction. Dai and Yuan [1999, 2003] present conjugate gradient schemes with the property that \( y_k^T d_k < 0 \) when \( y_k^T s_k > 0. \) In our algorithm observe that, if for all \( k, \) \( \theta_{k+1} \geq 1/4\delta_k, \) and the line search satisfies the Wolfe conditions, then for all \( k \) the search direction (18) and (20) satisfy the sufficient descent condition.

In Andrei [2006d] another algebraic interpretation of the conjugacy condition \( y_k^T d_{k+1} = 0 \) (where \( d_{k+1} \) is given as in (22)) is suggested as:
where \( Q_{k+1} \) is given in (24). After some algebra the following algorithm is obtained:

\[
d_{k+1} = -g_{k+1} + \beta_k^{ACGSD} s_k, \quad d_0 = -g_0,
\]

where

\[
\beta_k^{ACGSD} = \frac{1}{y_k^T s_k} \left( y_k - \frac{g_k^T y_k}{y_k^T s_k} s_k \right)^T g_{k+1}.
\]

With these the following variant of conjugate gradient with sufficient descent and conjugacy conditions can be presented:

**ACGSD Algorithm**

1. **Initialization.** Select \( x_0 \in \mathbb{R}^n \) and the parameters \( 0 < \sigma_1 < \sigma_2 < 1 \). Compute \( f(x_0) \) and \( g_0 \). Consider \( d_0 = -g_0 \) and \( \alpha_0 = 1/\|g_0\| \). Set \( k = 0 \).

2. **Test for continuation of iterations.** If \( \|g_k\|_\infty \leq 10^{-6} \), then stop, else set \( k = k + 1 \).

3. **Line search.** Compute \( \alpha_k \) satisfying the Wolfe line search conditions (4)–(5) and update the variables \( x_{k+1} = x_k + \alpha_k d_k \), \( g_{k+1} \), \( y_k = g_{k+1} - g_k \).

4. **Direction computation.** Compute \( d = -g_{k+1} + \beta_k^{ACGSD} s_k \), where \( \beta_k^{ACGSD} \) is computed as in (29). If \( g_{k+1}^T d \leq -10^{-3} \|d\|_2 \|g_{k+1}\|_2 \), then define \( d_{k+1} = d \), otherwise set \( d_{k+1} = -g_{k+1} \). Compute the initial guess \( \alpha_k = \alpha_{k-1} \|d_{k-1}\| / \|d_k\| \), set \( k = k + 1 \) and continue with step 2.

Another variant of this conjugate gradient algorithm with sufficient descent and conjugacy conditions is given by:

\[
\beta_k^{ACGSD_2} = \max \left\{ 0, \frac{g_{k+1}^T y_k}{y_k^T s_k} \left( 1 - \frac{g_{k+1}^T s_k}{y_k^T s_k} \right) \right\}.
\]

Finally, using the same arguments as above, but this time on the Polak-Ribi`ere-Polyak parameter \( \beta_k^{PRP} \) we get another conjugate gradient algorithm where

\[
\beta_k^{CGSD/PRP} = \frac{1}{y_k^T s_k} \left( y_k - \frac{\|y_k\|^2}{\|g_k\|^2} s_k \right)^T g_{k+1}.
\]

For all these algorithms some convergence results can be proved (see [Andrei 2006c, d]).
2.6. A Conjugate Gradient Algorithm with guaranteed descent and an efficient line search

Recently, for solving (1), Hager and Zhang [2004a, 2004b] presented a conjugate gradient algorithm with guaranteed descent and the performance of the Fortran 77 package CG_DESCENT which implements it. The directions \( d_k \) are computed by the following rule:

\[
d_{k+1} = -g_{k+1} + \beta_k d_k, \tag{32}
\]

\[
\beta_k = \frac{1}{d_k^T y_k} \left( y_k - 2 g_{k+1}^T d_k d_k^T y_k \right)^T g_{k+1}, \tag{33}
\]

\( d_0 = -g_0 \). In their algorithm Hager and Zhang restrict \( \beta_k \) to be nonnegative. This is motivated by the work of Gilbert and Nocedal [1992] who modified the Polak and Ribiére updating formula as \( \beta_k^+ = \max \{ \beta_k, 0 \} \) and proved the global convergence of this computational scheme for general nonlinear functions. Similar to the approaches considered by Gilbert and Nocedal [1992], Han, Liu, Sun and Yin [1994], and Wang, Han and Wang [2000] in their studies on the Polak-Ribiére version of the conjugate gradient method, Hager and Zhang prove the convergence for general nonlinear functions by restricting the lower bound of \( \beta_k \) in the following manner:

\[
d_{k+1} = -g_{k+1} + \bar{\beta}_k d_k, \tag{34}
\]

\[
\bar{\beta}_k = \max \{ \beta_k, \eta_k \}, \tag{35}
\]

\[
\eta_k = -\frac{1}{\|d_k\| \min \{ \eta, \|g_k\| \}}, \tag{36}
\]

where \( \beta_k \) is given by (33), and the parameter \( \eta > 0 \) is a user specified constant. (Suggested value: \( \eta = 0.01 \), considered in all numerical experiments).

An important innovation given by Hager and Zhang in their approach consists of a new efficient and highly accurate line search procedure. This is based on the Wolfe conditions (4) and (5) and on a very fine interpretation of the numerical issue concerning the first Wolfe condition (4).

3. Lennard-Jones Clusters. Molecular Conformation

The molecular conformation problem consists of determination of the minimum energy configuration of a cluster of atoms or molecules. This is a central problem in the study of cluster static [Hoare, 1979]. Given the positions \( p_1, p_2, \ldots, p_n \) of \( n \) atoms (points) in \( \mathbb{R}^3 \), the energy potential function is defined as
Conjugate gradient algorithms

$$V(p) = \sum_{j=2}^{n} \sum_{i=1}^{j-1} v(\|p_j - p_i\|_2),$$

where $\|\cdot\|$ denotes the Euclidean norm and $v : \mathbb{R} \to \mathbb{R}$ is the potential energy function between pairs of atoms. In classical molecular dynamics simulations the interaction between atoms in the sample are described with an interatomic potential $v(r)$, generally assumed to depend only on the distance $r$ between two atoms. Generally, the pairwise potential energy function satisfies the following requirements: it is twice continuously differentiable on $\mathbb{R}_+$, $v(1) = -1$, $(r - 1)v(r) > 0$ for $r \neq 1$, and

$$\int_{1}^{\infty} r^2 |v(r)| \, dr < \infty.$$

The most common choices for the interatomic potential are the Lennard-Jones and Morse potential. The Lennard-Jones is more suitable for closed shell systems such as the noble gases, while the Morse interatomic potential is more appropriate for metals [Lennard-Jones, 1931], [Morse, 1929].

The Lennard-Jones potential function we consider in this study is defined as

$$v(r) = r^{-12} - 2r^{-6}.$$

The Lennard-Jones potential function $v(.)$ is a non-convex function that is bounded below. Indeed, $v(r) \to +\infty$ as $r \to 0$, while $v(r)$ converges to zero as $r$ approaches $+\infty$. The global minimum of $v$ occurs at $r = 1$. The potential function $V$ is invariant with respect to permutations, translations and rotations of the $n$ atoms. Invariance with respect to translations can be eliminated by translating the cluster so that the center of gravity is at the origin.

The molecular conformation problem is to determine a configuration (position for the $n$ points in $\mathbb{R}^3$) such that the energy function $V$ is minimized. Therefore, we determine the configurations that are minimal in the sense that they correspond to a local minimizer of the molecular conformation problem with the least known value of the energy function. Of course, we get only a local minimum. Finding the global minimum of $V$ is a difficult problem because the number of local minima grows exponentially with the number of atoms $n$.

In the following we present the performances of 25 conjugate gradient algorithms on Lennard-Jones clusters, i.e. molecular conformation application (see MINPACK-2 collection: [Averick, Carter and Moré, 1991], [Averick, Carter, Moré, and Xue Guo-Liang, 1992]).

In this numerical study we consider only the performances of all these algorithms. All codes are written in Fortran 77 (default compiler settings) on a workstation Intel 4, 1.8 GHz. SCALCG is authored by Andrei, SCG by Birgin and Martínez [2001] and CG_DESCENT by Hager and Zhang [2004a]. The rest of algorithms are implemented by Andrei. In all these numerical experiments we have considered the standard initial point and the iterations were stopped according to the following criteria:
∥g_k∥_∞ ≤ 10^{-5} \text{ or } \alpha_k |g_k^T d_k| ≤ 10^{-20} |f(x_{k+1})|.

The SCALCG package uses the Powell restart criterion. SCG and scaled variants of Polak-Ribière-Polyak or Fletcher-Reeves use the angle restart criterion (see [Andrei, 2005]). All these algorithms use the same implementation of the Wolfe line search procedure with the same values of parameters \(\sigma_1\) and \(\sigma_2\).

Table 4 shows the results of the conjugate gradient algorithms considered in this study (\#iter is the number of iterations, \#fg is the number of function and its gradient evaluations and cpu(s) is the central processor time in seconds).

### Table 4. Performance of Conjugate Gradient Algorithms. Lennard-Jones Clusters. Molecular Conformation \(p_i \in \mathbb{R}^3\), natoms=1000, \(n = 3000\), Powell restart

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Algorithm</th>
<th>#iter</th>
<th>#fg</th>
<th>CPU</th>
<th>(f^\ast)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>HS (Hestenes-Stiefel)</td>
<td>1636</td>
<td>1988</td>
<td>169.15</td>
<td>-6603.499965</td>
</tr>
<tr>
<td>2.</td>
<td>FR (Fletcher-Reeves)</td>
<td>4001</td>
<td>5254</td>
<td>446.03</td>
<td>-5313.018443</td>
</tr>
<tr>
<td>3.</td>
<td>PRP (Polak-Ribière-Polyak)</td>
<td>3344</td>
<td>4632</td>
<td>392.91</td>
<td>-6629.091442</td>
</tr>
<tr>
<td>4.</td>
<td>PRP+ (Polak-Ribière-Polyak+)</td>
<td>1809</td>
<td>2554</td>
<td>216.70</td>
<td>-6607.397890</td>
</tr>
<tr>
<td>5.</td>
<td>CD (Conjugate Descent)</td>
<td>3152</td>
<td>3584</td>
<td>305.22</td>
<td>-6618.251234</td>
</tr>
<tr>
<td>6.</td>
<td>LS (Liu-Storey)</td>
<td>2129</td>
<td>2909</td>
<td>246.84</td>
<td>-6588.156816</td>
</tr>
<tr>
<td>7.</td>
<td>DY (Dai-Yuan)</td>
<td>4001</td>
<td>4015</td>
<td>343.08</td>
<td>-5245.489809</td>
</tr>
<tr>
<td>8.</td>
<td>DL(t=1) (Dai-Liao)</td>
<td>3427</td>
<td>4157</td>
<td>353.67</td>
<td>-6645.128435</td>
</tr>
<tr>
<td>9.</td>
<td>DL+(t=1) (Dai-Liao +)</td>
<td>1320</td>
<td>1594</td>
<td>135.61</td>
<td>-6605.079699</td>
</tr>
<tr>
<td>10.</td>
<td>aCGSD (Andrei)</td>
<td>1007</td>
<td>1148</td>
<td>97.75</td>
<td>-6628.423441</td>
</tr>
<tr>
<td>11.</td>
<td>hDY (hybrid Dai-Yuan)</td>
<td>1797</td>
<td>2175</td>
<td>185.03</td>
<td>-6624.322112</td>
</tr>
<tr>
<td>12.</td>
<td>hDYz (hybrid Dai-Yuan zero)</td>
<td>1402</td>
<td>1677</td>
<td>142.70</td>
<td>-6609.11685</td>
</tr>
<tr>
<td>13.</td>
<td>GN (Gilbert-Nocedal)</td>
<td>1559</td>
<td>2103</td>
<td>178.50</td>
<td>-6619.710430</td>
</tr>
<tr>
<td>14.</td>
<td>HuSt (Hu-Storey)</td>
<td>1345</td>
<td>1804</td>
<td>153.11</td>
<td>-6574.545172</td>
</tr>
<tr>
<td>15.</td>
<td>TAS (Touat-Ahmed-Storey)</td>
<td>2202</td>
<td>3007</td>
<td>255.14</td>
<td>-6613.138435</td>
</tr>
<tr>
<td>16.</td>
<td>LS-CD (hybrid LS-CD)</td>
<td>2333</td>
<td>3082</td>
<td>261.68</td>
<td>-6614.265345</td>
</tr>
<tr>
<td>17.</td>
<td>sP (Scaled Perry) (Birgin-Martínez)</td>
<td>1414</td>
<td>1660</td>
<td>141.34</td>
<td>-6600.920689</td>
</tr>
<tr>
<td>18.</td>
<td>sP+ (Scaled Perry+) (Birgin-Martínez+)</td>
<td>675</td>
<td>787</td>
<td>67.02</td>
<td>-6590.794367</td>
</tr>
<tr>
<td>19.</td>
<td>sPRP (scaled Polak-Ribière-Polyak)</td>
<td>2606</td>
<td>3555</td>
<td>301.68</td>
<td>-6629.087628</td>
</tr>
<tr>
<td>20.</td>
<td>sPR (scaled Fletcher-Reeves)</td>
<td>4001</td>
<td>5256</td>
<td>446.38</td>
<td>-5312.929909</td>
</tr>
<tr>
<td>21.</td>
<td>CGSD/PRP (Andrei)</td>
<td>2208</td>
<td>2672</td>
<td>227.34</td>
<td>-6591.471518</td>
</tr>
<tr>
<td>22.</td>
<td>ACGSD (Andrei)</td>
<td>813</td>
<td>915</td>
<td>77.97</td>
<td>-6621.009654</td>
</tr>
<tr>
<td>23.</td>
<td>ACGSDz (Andrei)</td>
<td>1066</td>
<td>1194</td>
<td>101.76</td>
<td>-6622.95559</td>
</tr>
</tbody>
</table>

For different number of atoms in \(\mathbb{R}^3\), the SCALCG package has the performances presented in Table 5. Table 6 shows the optimal value of the potential function \(V\) for different number of atoms.

Table 7 shows the performances of CG_DESCENT package by Hager and Zhang.
Table 5. Performance of SCALCG, $\theta_{k+1}$-spectral, Powell restart

<table>
<thead>
<tr>
<th>#atoms</th>
<th>n</th>
<th>#iter</th>
<th>#fg</th>
<th>cpu(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>3000</td>
<td>1252</td>
<td>2002</td>
<td>383.16</td>
</tr>
<tr>
<td>2000</td>
<td>6000</td>
<td>1019</td>
<td>1388</td>
<td>752.52</td>
</tr>
<tr>
<td>3000</td>
<td>9000</td>
<td>1931</td>
<td>2444</td>
<td>2984.55</td>
</tr>
<tr>
<td>4000</td>
<td>12000</td>
<td>3542</td>
<td>4452</td>
<td>9667.37</td>
</tr>
<tr>
<td>5000</td>
<td>15000</td>
<td>2642</td>
<td>3415</td>
<td>11590.63</td>
</tr>
</tbody>
</table>

Table 6. Optimal value of the potential function – SCALCG

<table>
<thead>
<tr>
<th>#atoms</th>
<th>n</th>
<th>$V^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>3000</td>
<td>-6602.762142</td>
</tr>
<tr>
<td>2000</td>
<td>6000</td>
<td>-13932.284675</td>
</tr>
<tr>
<td>3000</td>
<td>9000</td>
<td>-21135.347220</td>
</tr>
<tr>
<td>4000</td>
<td>12000</td>
<td>-28797.188265</td>
</tr>
<tr>
<td>5000</td>
<td>15000</td>
<td>-35902.028520</td>
</tr>
</tbody>
</table>

[2004a, 2004b] using the Wolfe line search and the approximate Wolfe line search, respectively.

Table 7. Performance of CG_DESCENT (tol=$10^{-5}$)

<table>
<thead>
<tr>
<th>n</th>
<th>#iter</th>
<th>#fg</th>
<th>cpu(s)</th>
<th>#iter</th>
<th>#fg</th>
<th>cpu(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000</td>
<td>1259</td>
<td>2586</td>
<td>631.65</td>
<td>1294</td>
<td>2526</td>
<td>619.58</td>
</tr>
<tr>
<td>6000</td>
<td>1721</td>
<td>3506</td>
<td>3393.11</td>
<td>1793</td>
<td>3480</td>
<td>3424.91</td>
</tr>
<tr>
<td>9000</td>
<td>1428</td>
<td>2924</td>
<td>6413.15</td>
<td>1494</td>
<td>2901</td>
<td>6496.64</td>
</tr>
</tbody>
</table>

In Table 8 we present the optimal value of the potential function $V$ for different number of atoms, as given by CG_DESCENT.

Table 8. Optimal value of the potential function – CG_DESCENT

<table>
<thead>
<tr>
<th>#atoms</th>
<th>n</th>
<th>$V^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>3000</td>
<td>-6615.4140837</td>
</tr>
<tr>
<td>2000</td>
<td>6000</td>
<td>-14082.394671</td>
</tr>
<tr>
<td>3000</td>
<td>9000</td>
<td>-21562.291265</td>
</tr>
</tbody>
</table>

We see that the scaled conjugate gradient algorithm SCALCG is top performer among these conjugate gradient algorithms. However, as we can see the optimal value given by CG_DESCENT is better than that obtained by SCALCG. This is due to the accurate implementation of the Wolfe line search.
4. Conclusion

In this study we have presented a survey and comparative numerical performances of a number of well known conjugate gradient algorithms for solving the molecular conformation problem. The conjugate gradient algorithms classify in four groups: conjugacy condition algorithms, hybrid conjugate gradient, scaled conjugate gradient and preconditioned conjugate gradient algorithms. All conjugate gradient algorithms are able to solve a large variety of large-scale unconstrained optimization problems including this difficult one which refers to the Lennard-Jones clusters problem. The above tables give computational evidence that our SCALCG - scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization is the top performer among the conjugate gradient algorithms. To get a conclusion some more numerical experiments must be considered, especially by comparing the conjugate gradient algorithms (mainly SCALCG and CG_DESCENT) with the limited memory BFGS (LBFGS by Nocedal) and truncated Newton (TN by Nash) algorithms.

References

Conjugate gradient algorithms


