Continuation Newton methods with deflation techniques and quasi-genetic evolution for global optimization problems

Xin-long Luo (luoxinlong@bupt.edu.cn)
Beijing University of Posts and Telecommunications https://orcid.org/0000-0002-4924-6007
Hang Xiao
Beijing University of Posts and Telecommunications

Research Article

Keywords: continuation Newton method, deflation technique, nonconvex optimization, global optimization, large-scale problem, genetic evolution

Posted Date: December 2nd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-1102775/v1

Journal of XXXX manuscript No.
(will be inserted by the editor)

Continuation Newton methods with deflation techniques and quasi-genetic evolution for global optimization problems

Xin-long Luo* · **Hang Xiao**

Received: date / Accepted: date

Abstract The global minimum point of an optimization problem is of interest in engineering fields and it is difficult to be solved, especially for a nonconvex large-scale optimization problem. In this article, we consider the continuation Newton method with the deflation technique and the quasi-genetic evolution for this problem. Firstly, we use the continuation Newton method with the deflation technique to find the stationary points from several determined initial points as many as possible. Then, we use those found stationary points as the initial evolutionary seeds of the quasi-genetic algorithm. After it evolves into several generations, we obtain a suboptimal point of the optimization problem. Finally, we use the continuation Newton method with this suboptimal point as the initial point to obtain the stationary point, and output the minimizer between this final stationary point and the found suboptimal point of the quasi-genetic algorithm. Finally, we compare it with the multi-start method (the built-in subroutine GlobalSearch.m of the MATLAB R2020a environment) and the differential evolution algorithm (the DE method, the subroutine de.m of the MATLAB Central File Exchange 2021), respectively. Numerical results show that the proposed method performs well for the large-scale global optimization problems, especially the problems of which are difficult to be solved by the known global optimization methods.

Keywords continuation Newton method · deflation technique · nonconvex optimization · global optimization · large-scale problem · genetic evolution

Mathematics Subject Classification (2010) 65K05 · 65L05 · 65L20

Xin-long Luo, Corresponding author
School of Artificial Intelligence, Beijing University of Posts and Telecommunications, P. O. Box 101, Xitucheng Road No. 10, Haidian District, 100876, Beijing China, E-mail: luoxinlong@bupt.edu.cn

Hang Xiao
School of Artificial Intelligence, Beijing University of Posts and Telecommunications, P. O. Box 101, Xitucheng Road No. 10, Haidian District, 100876, Beijing China, E-mail: xiaohang0210@bupt.edu.cn

1 Introduction

In this article, we are mainly concerned with the global minimum of the unconstrained optimization problem

$$\min_{x\in\Re^n} f(x), \tag{1}$$

where $f:\ \Re^n \to \Re$ is a differentiable function. When problem (1) is nonconvex and large-scale, it may have many local minimizers and it is challenge to find its global minimum. For problem (1), there are some popular global optimization methods, such as the multi-start methods [7, 15, 48, 56], the genetic evolution algorithms [20, 28, 40, 43] and their hybrid algorithms [22, 41, 51, 55].

For multi-start methods, they use the local minimum method such as the trust-region method [10, 14, 58] or the line search method [46, 53] to solve problem (1) and find its local minimum point x^*, which satisfies the following first-order optimality condition

$$F(x^*) = \nabla f(x^*) = 0. \tag{2}$$

Since the implementations of multi-start methods are simple and successful for many real-world problems, many commercial global solvers are based on these methods such as GlobalSearch.m of the MATLAB R2020a environment [42, 56]. However, for the multi-start method, a starting point could easily lie in the basin of attraction of a previously found local minimum point [48]. Hence, the algorithm might find the same local minimum even if the searches are initiated from points those are far apart.

The evolutionary algorithm [20, 45, 51, 52] is another popular method and used by a broad community of engineers and practitioners to solve real-world problems due to its simplicity and attractive nature-inspired interpretations. However, for the evolutionary algorithm, since there is not a guaranteed convergence and its computational time is very heavy for the large-scale problem, it easily fails to find the global minimum accurately.

In order to overcome the shortcomings of these two methods and make use of their advantages, we revise the continuation Newton method [4, 35, 38] by using the deflation technique [9] to find the stationary points of $f(x)$ as many as possible. Then, we select the first L smallest stationary points of $f(x)$ as the initial evolutionary seeds of the quasi-genetic algorithm [40, 43, 45]. After the quasi-genetic algorithm evolves into several generations such as twenty generations, we obtain a suboptimal point x^*_{ag} near the global minimum point. Finally, we use the continuation Newton method [35, 38] with this suboptimal point x^*_{ag} as the initial point to solve the nonlinear system (2) of equations and obtain one stationary point x^*_{cn} of $f(x)$. Finally, we output $x^*_{min} = \arg\min\{f(x^*_{ag}),\ f(x^*_{cn})\}$ as the global minimum point of the algorithm.

The rest of this article is organized as follows. In the next section, we convert the nonlinear system (2) into the continuation Newton flow. Then, we give the continuation Newton method with the trust-region updating strategy to follow the continuation Newton flow and obtain its steady-state solution, i.e. a stationary point of

$f(x)$. In section 3, we consider the deflation technique to construct a new nonlinear function $G_k(\cdot)$ such that the zeros of $G_k(x)$ are the stationary points of $f(x)$ and exclude the previously found stationary points $x_1^*, \ldots, x_k^*$ of $f(x)$. In section 4, we give a quasi-genetic algorithm to obtain the suboptimal point by using the evolutionary strategy of the convex combination and using these stationary points $x_1^*, \ldots, x_K^*$ as the evolutionary seeds. Then, we refine suboptimal solution by using the continuation Newton method from this suboptimal point to solve the system (2) of nonlinear equations. In section 5, some promising numerical results of the proposed method are reported, with comparison to the multi-start method (the subroutine GlobalSearch.m of the MATLAB R2020a environment) and the differential evolution algorithm (the subroutine de.m of the MATLAB Central File Exchange 2021 [57]). Finally, some discussions and conclusions are also given in section 6. $\|\cdot\|$ denotes the Euclidean vector norm or its induced matrix norm throughout the paper.

2 Finding a stationary point with the continuation Newton method

For the completeness of the paper, we restate the continuation Newton method with the trust-region updating strategy [35, 38] for finding a solution of the nonlinear system (2).

If we consider the damped Newton method for the nonlinear system (2) [25, 26, 47], we have

$$x_{k+1} = x_k - \alpha_k J(x_k)^{-1} F(x_k), \tag{3}$$

where $J(x_k)$ is the Jacobian matrix of $F(x_k)$. We regard $x_{k+1} = x(t_k + \alpha_k)$, $x_k = x(t_k)$ and let $\alpha_k \to 0$, then we obtain the following continuous Newton flow [6,8,11,35,54]:

$$\frac{dx(t)}{dt} = -J(x)^{-1} F(x), \quad x(t_0) = x_0. \tag{4}$$

Actually, if we apply an iteration with the explicit Euler method [5] for the continuous Newton flow (4), we obtain the damped Newton method (3). Since the Jacobian matrix $J(x) = F'(x)$ may be singular, we reformulate the continuous Newton flow (4) as the following general formula:

$$-J(x)\frac{dx(t)}{dt} = F(x), \quad x(t_0) = x_0. \tag{5}$$

The continuous Newton flow (5) is an old method and can be backtracked to Davidenko's work [11] in 1953. After that, it was investigated by Branin [8], Deuflhard et al [13], Tanabe [54] and Abbott [1] in 1970s, and applied to nonlinear boundary problems by Axelsson and Sysala [6] recently. The continuous and even growing interest in this method originates from its some nice properties. One of them is that the solution $x(t)$ of the continuous Newton flow (5) converges to the steady-state solution x^* from any initial point x_0, as described by the following property 2.1.

Property 2.1 (Branin [8] and Tanabe [54]) Assume that $x(t)$ is the solution of the continuous Newton flow (5), then $E(x(t)) = \|F(x)\|^2$ converges to zero when $t \to \infty$. That is to say, for every limit point x^* of $x(t)$, it is also a solution of the nonlinear system (2). Furthermore, every element $F^i(x)$ of $F(x)$ has the same convergence rate $exp(-t)$ and $x(t)$ can not converge to the solution x^* of the nonlinear system (2) on the finite interval when the initial point x_0 is not a solution of the nonlinear system (2).

Proof. Assume that $x(t)$ is the solution of the continuous Newton flow (5), then we have

$$\frac{d}{dt}\left(e^t F(x)\right) = e^t J(x)\frac{dx(t)}{dt} + e^t F(x) = 0.$$

Consequently, we obtain

$$F(x(t)) = F(x_0)e^{-t}. \tag{6}$$

From equation (6), it is not difficult to know that every element $F^i(x)$ of $F(x)$ converges to zero with the convergence rate e^{-t} when $t \to \infty$. Thus, if the solution $x(t)$ of the continuous Newton flow (5) belongs to a compact set, it has a limit point x^* when $t \to \infty$, and this limit point x^* is also a solution of the nonlinear system (2).

If we assume that the solution $x(t)$ of the continuous Newton flow (5) converges to the solution x^* of the nonlinear system (2) on the finite interval $(0, T]$, from equation (6), we have

$$F(x^*) = F(x_0)e^{-T}. \tag{7}$$

Since x^* is a solution of the nonlinear system (2), we have $F(x^*) = 0$. By substituting it into equation (7), we obtain

$$F(x_0) = 0.$$

Thus, it contradicts the assumption that x_0 is not a solution of the nonlinear system (2). Consequently, the solution $x(t)$ of the continuous Newton flow (5) can not converge to the solution x^* of the nonlinear system (2) on the finite interval. □

Remark 2.1 The inverse $J(x)^{-1}$ of the Jacobian matrix $J(x)$ can be regarded as the pre-conditioner of $F(x)$ such that the solution elements $x^i(t)\,(i = 1, 2, \ldots, n)$ of the continuous Newton flow (4) have the roughly same convergence rates and it mitigates the stiff property of the ODE (4) (the definition of the stiff problem can be found in [21] and references therein). This property is very useful since it makes us adopt the explicit ODE method to follow the trajectory of the Newton flow.

Actually, if we consider $F(x) = Ax$, from the ODE (5), we have

$$A\frac{dx}{dt} = -Ax,\ x(0) = x_0. \tag{8}$$

By integrating the linear ODE (8), we obtain

$$x(t) = e^{-t}x_0. \tag{9}$$

From equation (9), we know that the solution $x(t)$ of the ODE (8) converges to zero exponentially with the same rate e^{-t} when t tends to infinity.

When the Jacobian matrix $J(x)$ is singular or nearly singular, the ODE (5) is the system of differential-algebraic equations (DAEs) and its trajectory can not be efficiently followed by the general ODE method such as the backward differentiation formulas (the built-in subroutine ode15s.m of the MATLAB R2020a environment [5,21,42,49]). Thus, we need to construct the special method to handle this problem.

Since the continuous Newton flow (5) is intrinsically a nonlinear diminishing system for the energy function $f(x(t)) = \|F(x(t))\|^2$, it can be integrated by the strong stability preserving methods [17,18] and the steady-state solution x^* can be obtained after the long time integration. Here, we consider another continuation approach based on the traditional optimization methods for problem (5). We expect that the new method has the global convergence as the homotopy continuation methods and the fast convergence rate near the solution x^* as the merit-function methods. In order to achieve these two aims, we construct the special continuation Newton method with the new step size $\alpha_k = \Delta t_k/(1+\Delta t_k)$ [35–39] and the time step Δt_k is adaptively adjusted by the trust-region updating strategy for problem (5).

Firstly, we apply the implicit Euler method to the continuous Newton flow (5) [5,21], then we obtain

$$J(x_{k+1})\,(x_{k+1} - x_k) = -\Delta t_k F(x_{k+1}). \tag{10}$$

The scheme (10) is an implicit formula and it needs to solve a system of nonlinear equations at every iteration. To avoid solving the system of nonlinear equations, we replace the Jacobian matrix $J(x_{k+1})$ with $J(x_k)$ and substitute $F(x_{k+1})$ with its linear approximation $F(x_k) + J(x_k)(x_{k+1} - x_k)$ into equation (10), respectively. Thus, we obtain the following explicit continuation Newton method:

$$J(x_k)s_k^N = -F(x_k),\ s_k = \frac{\Delta t_k}{1+\Delta t_k}s_k^N,\ x_{k+1} = x_k + s_k. \tag{11}$$

In order to improve the robustness of the algorithm, we use the QR factorization (pp. 246-249 in [19]) to solve the linear system (11) as follows:

$$J_k^T = Q_k R_k,\ R_k^T d_k = -F_k,\ s_k^N = Q_k d_k,\ s_k = \frac{\Delta t_k}{1+\Delta t_k}s_k^N,\ x_{k+1} = x_k + s_k,$$

where $J_k = J(x_k)$, $F_k = F(x_k)$ and Q_k is an $n \times n$ orthogonal matrix.

Remark 2.2 The continuation Newton method (11) is similar to the damped Newton method (3) if we let $\alpha_k = \Delta t_k/(1+\Delta t_k)$ in equation (11). However, from the view of the ODE method, they are different. The damped Newton method (3) is obtained by the explicit Euler method applied to the continuous Newton flow (5), and its time-stepping size α_k is restricted by the numerical stability [21]. That is to say,

for the linear test equation $dx/dt = -\lambda x$, its time step size α_k is restricted by the stable region $|1-\lambda\alpha_k| \leq 1$. Therefore, the large time step α_k can not be adopted in the steady-state phase. The continuation Newton method (11) is obtained by the implicit Euler method and its linear approximation applied to the continuous Newton flow (5), and its time step size Δt_k is not restricted by the numerical stability for the linear test equation. Therefore, the large time step can be adopted in the steady-state phase for the continuation Newton method (11), and it mimics the Newton method near the solution x^* such that it has the fast local convergence rate. The most of all, $\alpha_k = \Delta t_k/(\Delta t_k + 1)$ in equation (11) is favourable to adopt the trust-region updating strategy for adaptively adjusting the time step Δt_k such that the continuation Newton method (11) accurately traces the trajectory of the continuous Newton flow in the transient-state phase and achieves the fast convergence rate near the equilibrium point x^*.

For a real-world problem, the analytical Jacobian $J(x_k)$ may not be offered. Thus, in practice, we replace the Jacobian matrix $J(x_k)$ with its difference approximation as follows:

$$J(x_k) \approx \left[\frac{F(x_k+\varepsilon e_1)-F(x_k)}{\varepsilon}, \ldots, \frac{F(x_k+\varepsilon e_n)-F(x_k)}{\varepsilon}\right], \tag{12}$$

where e_i represents the unit vector whose elements equal zeros except for the i-th element which equals 1, and the parameter ε can be selected as 10^{-6} according to our numerical experiments.

Another issue is how to adaptively adjust the time step Δt_k at every iteration. There is a popular way to control the time step size based on the trust-region updating strategy [12, 23, 29, 31–34, 36]. Its main idea is that the time step size Δt_{k+1} will be enlarged when the linear model $F(x_k)+J(x_k)s_k$ approximates $F(x_k+s_k)$ well, and Δt_{k+1} will be reduced when $F(x_k)+J(x_k)s_k$ approximates $F(x_k+s_k)$ badly. Thus, by using the relation (11), we enlarge or reduce the time step size Δt_k at every iteration according to the following ratio:

$$\rho_k = \frac{\|F(x_k)\| - \|F(x_k+s_k)\|}{\|F(x_k)\| - \|F(x_k)+J(x_k)s_k\|} = \frac{\|F(x_k)\| - \|F(x_k+s_k)\|}{(\Delta t_k/(1+\Delta t_k))\|F(x_k)\|}. \tag{13}$$

A particular adjustment strategy is given as follows:

$$\Delta t_{k+1} = \begin{cases} 2\Delta t_k, & \text{if } 0 \leq |1-\rho_k| \leq \eta_1, \\ \Delta t_k, & \text{else if } \eta_1 < |1-\rho_k| < \eta_2, \\ \frac{1}{2}\Delta t_k, & \text{others}, \end{cases} \tag{14}$$

where the constants are selected as $\eta_1 = 0.25,\ \eta_2 = 0.75$ according to our numerical experiments. When $\rho_k \geq \eta_a$, we accept the trial step and set

$$s_k = \frac{\Delta t_k}{1+\Delta t_k} s_k^N,\ x_{k+1} = x_k + s_k, \tag{15}$$

where the Newton step s_k^N is solved by equation (11) and η_a is a small positive number such as $\eta_a = 1.0 \times 10^{-6}$. Otherwise, we discard the trial step and set

$$x_{k+1} = x_k,\ s_{k+1}^N = s_k^N,\ F_{k+1} = F_k. \tag{16}$$

Remark 2.3 This new time-stepping selection based on the trust-region updating strategy has some advantages compared to the traditional line search strategy [30]. If we use the line search strategy and the damped Newton method (3) to track the trajectory $x(t)$ of the continuous Newton flow (5), in order to achieve the fast convergence rate in the steady-state phase, the time step α_k of the damped Newton method is tried from 1 and reduced by the half with many times at every iteration. Since the linear model $F(x_k) + J(x_k)s_k^N$ may not approximate $F(x_k + s_k^N)$ well in the transient-state phase, the time step α_k will be small. Consequently, the line search strategy consumes the unnecessary trial steps in the transient-state phase. However, the time-stepping selection method based on the trust-region updating strategy (13)-(14) can overcome this shortcoming.

According to the above discussions, we give the detailed descriptions of the continuation Newton method and the trust-region updating strategy for finding a stationary point of $f(x)$ (i.e. a solution of the nonlinear system (2)) in Algorithm 1. The global convergence and local suplinear convergence of Algorithm 1 can be found in [35, 38].

3 The deflation technique

By using Algorithm 1 (i.e., the CNMTr method) in section 2, we can find a stationary point x_1^* of $f(x)$. Our strategy is to use Algorithm 1 repeatedly, until the stationary points $x_k^*(k = 1, 2, \ldots, K)$ of $f(x)$ are found as many as possible. Then, we select the first L minimizers of $f(x)$ from the set $\{x_1^* \ldots, x_K^*\}$ as the initial seeds of the genetic algorithm in section 4.

If we directly use Algorithm 1 and the multi-start techniques, a starting point could easily lie in the basin of attraction of a previously found local minimum and algorithm 1 might find the same local minimum even if the searches are initiated from points those are far apart. In order to overcome this disadvantage, we use the deflation technique [9] described as follows.

Assume that x_k^* is a zero point of $G_{k-1}(x)$. Then, we construct another function $G_k(\cdot)$ via eliminating the zero point x_k^* of $G_{k-1}(x)$ as follows:

$$G_k(x) = \frac{1}{\|x - x_k^*\|} G_{k-1}(x), \tag{17}$$

where $G_0(x) = F(x) = \nabla f(x)$. Thus, from equation (17), we obtain

$$G_k(x) = \frac{1}{\|x - x_1^*\| \cdots \|x - x_k^*\|} F(x), \tag{18}$$

Algorithm 1 Continuation Newton methods and the trust-region updating strategy for finding a solution of nonlinear equations (The CNMTr method)

Input:

Function $F:\ \Re^n \to \Re^n$, the initial point x_0 (optional), the Jacobian J of F (optional), the tolerance ε (optional).

Output:

An approximation solution x^* of nonlinear equations.

Set the default $x_0 = \text{ones}(n,1)$ and $\varepsilon = 10^{-6}$ when x_0 or ε is not provided by the calling subroutine.
Initialize the parameters: $\eta_a = 10^{-6},\ \eta_1 = 0.25,\ \gamma_1 = 2,\ \eta_2 = 0.75,\ \gamma_2 = 0.5,\ \text{maxit} = 400$.
Set $\Delta t_0 = 2$, flag_success_trialstep = 1, itc $= 0,\ k = 0$.
while (itc $<$ maxit) **do**
  if (flag_success_trialstep == 1) **then**
    Set itc = itc + 1.
    Compute $\text{Res}_k = \|F_k\|_\infty$.
    if ($\text{Res}_k < \varepsilon$) **then**
      break;
    end if
    Evaluate $F_k = F(x_k)$ and $J_k = J(x_k)$.
    By using the qr decomposition $[Q_k, R_k] = \text{qr}(J_k^T)$ of J_k^T, we obtain the orthogonal matrix Q_k and the upper triangle matrix R_k.
    By solving $R_k^T d_k = -F_k$ and $s_k^N = Q_k d_k$, we obtain the Newton step s_k^N.
  end if
  Set $s_k = \Delta t_k/(1+\Delta t_k)s_k^N,\ x_{k+1} = x_k + s_k$.
  Evaluate $F_{k+1} = F(x_{k+1})$.
  if ($\|F_k\| < \|F_{k+1}\|$) **then**
    $\rho_k = -1$;
  else
    Compute the ratio ρ_k from equation (13).
  end if
  Adjust the time-stepping size Δt_{k+1} according to the trust-region updating strategy (14).
  if ($\rho_k \geq \eta_a$) **then**
    Accept the trial point x_{k+1}. Set flag_success_trialstep = 1.
  else
    Set $x_{k+1} = x_k,\ F_{k+1} = F_k,\ s_{k+1}^N = s_k^N$, flag_success_trialstep = 0.
  end if
  Set $k \longleftarrow k+1$.
end while

where $x_i^*\ (i = 1:k)$ are the zero points of $F(x)$.

When $F(x_i^*) = 0$ and $J(x_i^*)$ is nonsingular, Brown and Gearhard [9] have proved

$$\liminf_{j\to\infty} \|G_k(x_j)\| > 0$$

for any sequence $x_j \to x_i^*$, where $G_k(x)$ is defined by equation (18). Actually, we can obtain the following estimation of the lower bound of $G_k(x)$ when x belongs to the neighborhood of x_i^*.

Lemma 3.1 *Assume that x_i^* is a zero point of the continuously differential function $F(\cdot)$ and its Jacobian matrix $J(x_i^*)$ satisfies*

$$\|J(x_i^*)y\| \geq c_l\|y\|,\ \forall\ y \in R^n. \tag{19}$$

Then there exists a neighborhood $B_{\delta_i}(x_i^*)$ of x_i^* such that

$$\|G_k(x)\| \geq \frac{c_l}{2} \frac{1}{\|x-x_1^*\|\cdots\|x-x_{i-1}^*\|\|x-x_{i+1}^*\|\cdots\|x-x_k^*\|}, \ \forall x \in B_{\delta_i}(x_i^*) \tag{20}$$

holds for all $x \in B_{\delta_i}(x_i^*)$, where $G_k(x)$ is defined by equation (18). That is to say, x_i^* is not the zero point of $G_k(x)$.

Proof. Since x_i^* is a zero point of $F(x)$, according to the first-order Taylor expansion, we have

$$\begin{aligned} F(x) &= F(x_i^*) + \int_0^1 J(x_i^* + t(x-x_i^*))(x-x_i^*)dt \\ &= \int_0^1 J(x_i^* + t(x-x_i^*))(x-x_i^*)dt, \end{aligned} \tag{21}$$

where $J(x) = F'(x)$ is the Jacobian matrix of $F(x)$. According to the nonsingular assumption (19) of $J(x_i^*)$ and the continuity of $J(\cdot)$, we know that there exists a neighborhood $B_{\delta_i}(x_i^*) = \{x : \|x-x_i^*\| \leq \delta_i\}$ of x_i^* such that

$$\|J(x) - J(x_i^*)\| \leq \frac{c_l}{2} \tag{22}$$

holds for all $x \in B_{\delta_i}(x_i^*)$. From equations (21)-(22), we have

$$\begin{aligned} &\|F(x) - J(x_i^*)(x-x_i^*)\| = \left\| \int_0^1 (J(x_i^* + t(x-x_i^*)) - J(x_i^*))(x-x_i^*)dt \right\| \\ &\leq \int_0^1 \|J(x_i^* + t(x-x_i^*)) - J(x_i^*)\|\|x-x_i^*\|dt \leq \frac{c_l}{2}\|x-x_i^*\|, \ \forall x \in B_{\delta_i}(x_i^*). \end{aligned} \tag{23}$$

Furthermore, from the triangle inequality $\|x-y\| \geq \|x\| - \|y\|$ and the assumption (19), we have

$$\|F(x) - J(x_i^*)(x-x_i^*)\| \geq \|J(x_i^*)(x-x_i^*)\| - \|F(x)\| \geq c_l\|x-x_i^*\| - \|F(x)\|. \tag{24}$$

Thus, from equations (23)-(24), we obtain

$$\|F(x)\| \geq \frac{c_l}{2}\|x-x_i^*\|, \ \forall x \in B_{\delta_i}(x_i^*). \tag{25}$$

By substituting inequality (25) into equation (18), we obtain

$$\|G_k(x)\| \geq \frac{c_l}{2} \frac{1}{\|x-x_1^*\|\cdots\|x-x_{i-1}^*\|\|x-x_{i+1}^*\|\cdots\|x-x_k^*\|} > 0, \ \forall x \in B_{\delta_i}(x_i^*).$$

That is to say, x_i^* is not a zero point of $G_k(x)$. □

In practice, we need to avoid the overflow or the underflow when we compute $G_k(x)$ and expect that the continuation Newton method (Algorithm 1) can find the

zero points of $F(x)$ as many as possible. Thus, according to our numerical experiments, we modify $G_k(x)$ defined by equation (17) as

$$G_k(x) = \frac{\alpha_k}{\|x - x_k^*\|_1} G_{k-1}(x), \tag{26}$$

where $G_0(x) = F(x)$ and α_k is defined by

$$\alpha_k = \begin{cases} n, \text{ if } \|x_k^*\|_1 \leq 10^{-6}, \\ \|x_k^*\|_1, \text{ otherwise.} \end{cases} \tag{27}$$

Since $G_k(x)$ has the special structure and its sub-differential exists except for finite points $x_1^*, \ldots, x_k^*$, from equations (26)-(27), we can compute its sub-differential $\partial G_k(x)$ of $G_k(x)$ as follows:

$$\begin{aligned} \partial G_k(x) &= \frac{\alpha_1 \cdots \alpha_k}{\|x - x_1^*\|_1 \cdots \|x - x_k^*\|_1} \frac{\partial F(x)}{\partial x} + \frac{\alpha_1 \cdots \alpha_k}{\|x - x_1^*\|_1 \cdots \|x - x_k^*\|_1} F(x) p(x)^T \\ &= \frac{\alpha_1 \cdots \alpha_k}{\|x - x_1^*\|_1 \cdots \|x - x_k^*\|_1} \left(\frac{\partial F(x)}{\partial x} + F(x) p(x)^T \right), \end{aligned} \tag{28}$$

where the column vector function $p(\cdot)$ is defined by

$$p(x)^T = -\left(\frac{\operatorname{sgn}(x - x_1^*)}{\|x - x_1^*\|_1} + \cdots + \frac{\operatorname{sgn}(x - x_k^*)}{\|x - x_k^*\|_1} \right), \tag{29}$$

the sign function $\operatorname{sgn}(\cdot)$ of which is defined by

$$\operatorname{sgn}(\beta) = \begin{cases} 1, \text{ if } \beta > 0, \\ 0, \text{ if } \beta = 0, \\ -1, \text{ otherwise.} \end{cases} \tag{30}$$

Now, by using the strategy (26) of eliminating zero points of $F(x)$ repeatedly, we can find the most of stationary points of $f(x)$. Then, we select the minimizer from the stationary points of $f(x)$ as the approximation of the global minimizer of problem (1). We describe its details in Algorithm 2.

Remark 3.1 From equation (26), we know that $F(x_l)$ may converge to a positive number when $\lim_{l\to\infty} G_k(x_l) = 0$ and $\lim_{l\to\infty} \|x_l - x_i^*\| = \infty$. One of the reasons is that the Newton method converges to the same solution of $F(x)$ and can not converge to other solutions of $F(x)$ when it is initialized in the special region [1]. Thus, the continuation Newton method with the deflation technique may not find all the stationary points of $f(x)$ from the same initial point x_0. In order to find the stationary point of $f(x)$ as many as possible, we use Algorithm 2 to find its stationary points from several far apart initial points sequentially such as $x_0 = \text{ones}(n,1)$, $x_0 = [\text{ones}(\frac{n}{2},1); -\text{ones}(\frac{n}{2},1)]$, $x_0 = [-\text{ones}(\frac{n}{2},1); \text{ones}(\frac{n}{2},1)]$ and $x_0 = -\text{ones}(n,1)$. According to our numerical experiments, the continuation Newton method with the deflation technique (i.e. Algorithm 2) can find all the stationary points of $f(x)$ when the number of its stationary points is less than thirty and Algorithm 2 is initialized from those four points in the most cases.

Algorithm 2 Continuation Newton methods with the deflation technique for multi-solutions of nonlinear equations (The CNMDT method)

Input:
The gradient $F(x) = \nabla f(x)$ of $f(x)$, the four initial points $x_0 = \text{ones}(n,1)$, $x_0 = [\text{ones}(n/2,1); -\text{ones}(n/2,1)]$, $x_0 = [-\text{ones}(n/2,1); \text{ones}(n/2,1)]$ and $x_0 = -\text{ones}(n,1)$, the tolerance ε (optional), the Jacobian $J(x)$ of $F(x)$ (optional).

Output:
The K approximation zero points $x_1^*, \cdots, x_K^*$ of $F(\cdot)$ and the minimizer x_{ap}^* of $f(x)$.

Set Number_of_Stationary_Points = 0.
Call Algorithm 1 to find a zero of F and we denote it as x_1^*.
Set Number_of_Stationary_Points = Number_of_Statonary_Points +1.
Set $G_0(x) = F(x)$.
Set Flag_of_Fail = 0 and $k = 0$.
while (Flag_of_Fail == 0) **do**
    Call Algorithm 1 to find a zero of $G_{k+1}(\cdot)$ and we denote it as x_{k+2}^*, where $G_{k+1}(\cdot)$ is defined by equations (26)-(27).
    if ("A zero of G_{k+1} is obtained by Algorithm 1 successfully.") **then**
        Set Flag_of_Fail = 0.
        Set Number_of_Stationary_Points = Number_of_Stationary_Points +1.
    else
        Set Flag_of_Fail = 1.
    end if
    Try again starting at step 6 with $k \longleftarrow k+1$.
end while
Set $f_{ap} = f(x_1^*)$ and $x_{ap}^* = x_1^*$.
for k = 2 : Number_of_Stationary_Points **do**
    if $f(x_k^*) < f_{ap}$ **then**
        Set $f_a = f(x_k^*)$ and $x_{ap}^* = x_k^*$.
    end if
end for

4 The quasi-genetic evolution and the local search method

By using the continuation Newton method with the deflation technique (Algorithm 2) in section 3, we can find the most of stationary points of $f(x)$ and the suboptimal minimizer x_{ap}^* of $f(x)$ when it has infinite stationary points. According to our numerical experiments, the global minimizer is one of the found stationary points or in the convex region generated by the found stationary points for the most problems. Therefore, we use the idea of memetic algorithms [45,51] to approach the global minimum further. Namely, we use the heuristic crossover of the genetic algorithm to evolve for approaching the global minimum of $f(x)$ from its stationary points, which are solved by Algorithm 2. We describe it in detail as follows.

Firstly, we use Algorithm 2 to find the stationary points of $f(x)$ as many as possible, and we denote these stationary points as $x_k^*\,(k = 1, 2, \ldots, K)$. Then, we select the first L minimizers of $f(x)$ from the set $\{x_1^*, \ldots, x_K^*\}$ and denote these L stationary points as $x_k^0\,(k = 1, 2, \ldots, L)$. We set $S_0 = \{x_1^0, \ldots, x_L^0\}$ and adopt them as the initial seeds of the quasi-genetic algorithm.

Secondly, we select any two elements x_i^0 and x_j^0 from the seed set S_0, and use the convex crossover to generate next offspring. Namely, we set

$$\bar{x}_k^1 = \frac{1}{2}\left(x_i^0 + x_j^0\right),\ i, j = 1:L,\ k = 1:\frac{1}{2}L(L-1). \tag{31}$$

Similarly, we select the first L minimizers of $f(x)$ from the set $S_0 \bigcup \left\{\bar{x}_1^1, \ldots, \bar{x}_{L(L-1)/2}^1\right\}$ and denote these first L minimizers as x_k^1 $(k = 1, \ldots, L)$. We set $S_1 = \left\{x_1^1, \ldots, x_L^1\right\}$ and regard them as the good population.

Repeatedly, after the M-th evolution, we generate the offspring $\left\{\bar{x}_1^M, \ldots, \bar{x}_{L(L-1)/2}^M\right\}$ and select the first L minimizers of $f(x)$ from the set $S_{M-1} \bigcup \left\{\bar{x}_1^M, \ldots, \bar{x}_{L(L-1)/2}^M\right\}$. We denote these first L minimizers as x_k^M $(k = 1, \ldots, L)$, set $S_M = \left\{x_1^M, \ldots, x_L^M\right\}$ and stop evolving. We denote the minimizer of $f(x)$ in the set S_M as x_{ag}^*.

Finally, in order to improve the convergence of the quasi-genetic evolution, we use the continuation Newton method (i.e. Algorithm 1) from the initial point x_{ag}^* to find the solution of $F(x)$ and denote this solution as x_{cn}^*, where we set $F(x) = \nabla f(x)$ in Algorithm 1. We output the minimum point x_{min}^* of $f(x)$ between x_{cn}^* and x_{ag}^*, i.e.

$$x_{min}^* = \arg\min\left\{f(x_{cn}^*),\ f(x_{ag}^*)\right\}.$$

We can select the parameter $L = \min\{20,\ K\}$ and $M = 10$ according to our numerical experiments. Now, we give the detailed descriptions of the continuation Newton method with the deflation technique and the quasi-genetic algorithm for the global optimization problem in Algorithm 3.

5 Numerical Experiments

In this section, some numerical experiments are conducted to test the performance of the continuation Newton method with the deflation technique and the quasi-genetic evolution (i.e. Algorithm 3, the CNMGE method) for the global optimization problems. The codes are executed by a HP notebook with the Intel quad-core CPU and 8Gb memory. We compare CNMGE with the multi-start algorithm (the built-in subroutine GlobalSearch.m of the MATLAB R2020a environment [42, 56]) and the differential evolution algorithm (the DE method, the subroutine de.m of the MATLAB Central File Exchange 2021 [20, 42, 57]) for 68 unconstrained global optimization problems which can be found in [2, 3, 16, 27, 44, 50].

GlobalSearch is a stable and efficient global optimization solver and is widely used in engineering field. DE is a modern evolutionary algorithm for solving small-scale global optimization effectively. Therefore, we select these two methods as the basis for comparison. We set the parameters of DE in line with the parameters of the evolutionary algorithm in [51, 52]. Namely, for small-scale problems, the population size is set to 50, the number of evolution generations is set to 1000, and the crossover

Algorithm 3 Continuation Newton methods with the deflation technique and the quasi-genetic evolution for the global optimization problem (the CNMGE method)

Input:

The gradient $F(x) = \nabla f(x)$ of $f(x)$, the initial point x_0 (optional), the tolerance ε (optional), the Jacobian $J(x)$ of $F(x)$ (optional).

Output:

An approximation minimizer x^*_{min} of $f(x)$.

1: Call Algorithm 2 to find the stationary points of $f(x)$ as many as possible and denote these stationary points as x^*_k $(k = 1, 2, \ldots, K)$.
2: Set $L = \min\{20,\ K\}$, $l = 0$ and the maximum evolution iterations $M = 10$.
3: Select the first L minimizers of $f(x)$ from the set $\{x^*_1 \ldots, x^*_K\}$ and denote these first L minimizers as x^0_k $(k = 1, 2, \ldots, L)$. Set $S_0 = \{x^0_1, \ldots, x^0_L\}$.
4: **while** $(l < M)$ **do**
5: Set $k = 1$.
6: **for** $i = 1 : L$ **do**
7: **for** $j = i+1 : L$ **do**
8: Set $\bar{x}^{l+1}_k = \frac{1}{2}\left(x^l_i + x^l_j\right)$;
9: Set $k = k+1$;
10: **end for**
11: **end for**
12: Select the first L minimizers of $f(x)$ from the set $S_l \bigcup \left\{\bar{x}^{l+1}_1, \ldots, \bar{x}^{l+1}_{L(L+1)/2}\right\}$, and denote these first L minimizers as x^{l+1}_i $(i = 1, \ldots, L)$. Set $S_{l+1} = \left\{x^{l+1}_1, \ldots, x^{l+1}_L\right\}$.
13: Set $l = l+1$.
14: **end while**
15: Select the smallest minimizer x^*_{ag} of $f(x)$ from S_M.
16: Set $F(x) = \nabla f(x)$ and call Algorithm 1 with the initial point x^*_{ag} to find the solution x^*_{cn} of $F(x)$.
17: Set $x^*_{min} = \arg\min\left\{f(x^*_{cn}),\ f(x^*_{ag})\right\}$.

probability is 0.2. For large-scale problems, the population size is set to 300, the number of evolution generations is set to 5000, and the crossover probability is set to 0.2. The termination condition of Algorithm 1 for finding a solution of the nonlinear system (2) is

$$\|F(x_k)\|_\infty \leq 10^{-6}. \tag{32}$$

The scales of test problems vary from dimension 1 to 1000. The numerical results are arranged in Tables 1-2. Since the global minima of some test problems are unknown, we mark its global minima of these problems with the notation "\" in Table 1 and Table 2. For these problems, we regard that the method fails to find their global minima if its found solutions are greater than the minimum solutions. Due to the randomness of DE, we repeatedly calculate every problem ten times with DE, and mark its average value and its minimum with "avg $f(x)$" and "min $f(x)$" in Tables 1-2, respectively. "Num. Loc. Sol." represents the number of the stationary points found by CNMGE in Tables 1-2. The computational time of CNMGE, GlobalSearch and DE is illustrated by Figures 1-2.

From Tables 1-2, we find that CNMGE performs well for unconstrained global optimization problems and finds their global minima. GlobalSearch fails to find the

global minima of test problems about one quarter. DE performs well for small-scale global optimization problems. However, it can not effectively solve large-scale global optimization problems. Therefore, CNMGE is more robust than GlobalSearch and DE to find the global minimum of the unconstrained optimization problem.

From Table 1 and Figure 1, we find that the computational time of CNMGE is slightly more than that of GlobalSearch for the large-scale unconstrained optimization problem. However, according to Table 1, the computational time of CNMGE is not serious for the large-scale unconstrained optimization problem, since it only consumes several seconds to one hundred and thirty seconds for large-scale unconstrained optimization problems.

From Table 2 and Figure 2, we find that the computational time of CNMGE is about $1/5$ of that of GlobalSearch or DE for the small-scale unconstrained optimization problem. One of the reasons is that the large-scale unconstrained optimization problem has more local minima than the small-scale unconstrained optimization problem and the cost of solving large-scale unconstrained optimization problems is much higher than that of small-scale unconstrained optimization problems. Thus, CNMGE consumes the more time to find the stationary points of the large-scale unconstrained optimization problem than that of the small-scale unconstrained optimization problem.

When the unconstrained optimization problem (1) has many local minima, the numerical tests show that Algorithm 2 can find at least 30 stationary points of $f(x)$, which are useful to make Algorithm 3 find the global minimum of $f(x)$ for the difficult problem. This result can be illustrated by Problem 1. Problem 1 [27] is a molecular potential energy problem and its objective function has the following form:

$$f(\omega) = \sum_{i=1}^{n} \left(1 + cos(3\omega_i) + \frac{(-1)^i}{\sqrt{10.60099896 - 4.141720682(cos\omega_i)}} \right), \quad (33)$$

where ω_i is the torsion angle and $n+3$ is the number of atoms or beads in the given system. The number of its local minimizers increases exponentially with the size of the problem, which characterizes the principal difficult in minimizing molecular potential energy functions for the traditional global optimization methods. CNMGE can find 91 stationary points of problem (33) with $n = 1000$ and its global minimum $f(\omega^*) = -41.118303$ at $\omega^* = (1.039195, 3.141593, \ldots, 1.039195, 3.141593)$. However, for problem 1 with 50 atoms, the traditional global optimization methods such as the interval analysis method [24, 27] or the multi-start method [42, 56] are difficult to find its global minimum. The differential evolution method (DE) also fails to solve this problem.

For problem 44, reference [50] gives its global minimum value (i.e. $-9.596407E+02$) at $x = (512, 404.2319)$, when it is evaluated on the interval $x_i \in [-512, 512]$ $(i = 1, 2)$. Since these three methods do not consider the constraints $x_i \in [-512, 512]$ $(i = 1, 2)$ and regard this problem as an unconstrained optimization problem, CNMGE can find a smaller value of $f(x)$ than that of reference [50].

Table 1: Numerical results of large-scale unconstrained optimization problems.

Problem	CNMGE		GlobalSearch.m		DE.m		Global Minimum
	CPU time (s) (Num. Loc. Sol.)	min $f(x)$	CPU time (s)	min $f(x)$	CPU time (s)	avg $f(x)$ min $f(x)$	
1. Molecular Energy Problem (n = 1000) [27]	8.94E+01 (91)	-41.1183	1.82E+01 (failed)	-0.2876	5.11E+02 (failed)	8.1144E+02 8.0396E+02	-41.1183
2. Ackley Function (n = 1000) [50]	4.98E+01 (27)	4.3552E-07	1.92E+01	8.8817E-16	2.01E+02 (failed)	13.6242 13.5875	0
3. Levy Function (n = 1000) [50]	1.72E+01 (9)	5.0587E-15	4.55 (failed)	0.3651	1.98E+02 (failed)	1.1500E+04 1.1224E+04	0
4. Schwefel Function (n = 1000) [50]	4.96E+01 (63)	0.01273	4.36E+01 (failed)	4.7903E+03	2.37E+02 (failed)	2.9806E+05 2.9450E+05	0.0127
5. Rastrigin Function (n = 1000) [50]	28.05 (46)	0	3.47	0	1.98E+02 (failed)	1.5100E+04 1.5025E+04	0
6. Styblinski-Tang Function (n = 1000) [50]	91.33 (65)	-3.9166E+04	4.52 (failed)	-2.5013E+04	4.0194E+02 (failed)	-6.8858E+03 -8.1869E+03	-3.9166E+04
7. Trid Function (n = 1000) [50]	2.34 (1)	-1.6716E+08	4.34 (failed)	-2.4920E+04	1.55E+02 (failed)	2.6463E+14 2.6009E+14	-1.6716E+08
8. Sum Squares Function (n = 1000) [50]	3.39 (1)	1.2528E-15	3.84	0	1.52E+02 (failed)	7.6699E+06 7.5501E+06	0
9. Sphere Function (n = 1000) [50]	0.57 (1)	0	3.22	3.9800E-11	1.50E+02 (failed)	5.0143E+03 4.8773E+03	0
10. Rotated Hyper-Ellipsoid Function (n = 1000) [50]	3.56 (1)	1.2528E-15	15.39	0	1.47E+02 (failed)	5.4078E+03 5.3586E+03	0
11. Zakharov Function (n = 1000) [50]	7.69 (1)	7.9095E-23	5.17	0	1.55E+02 (failed)	2.3905E+04 2.3363E+04	0
12. Dixon-Price Function (n = 1000) [50]	13.05 (35)	2.6176E-16	4.03 (failed)	0.7293	1.52E+02 (failed)	2.0240E+09 1.9619E+09	0
13. Rosenbrock Function (n = 1000) [44, 50]	5.25 (1)	2.5439E-09	3.82 (failed)	30.6300	1.64E+02 (failed)	2.1546E+07 2.0745E+07	0
14. Powell Function (n = 1000) [44, 50]	1.29E+02 (56)	2.5807E-10	4.72	0	2.88E+02 (failed)	1.2364E+06 1.1879E+06	0
15. Quartic With Noise Function (n = 1000) [3]	6.07 (11)	0.5000	3.88	0.5000	3.9642E+02 (failed)	4.9558E+04 1.8505E+04	0.5000
16. Schubert Function (n = 1000) [3]	89.16 (85)	-1.2031E+04	62.95 (failed)	-9.4947E+03	3.26E+02 (failed)	-5.0650E+02 -5.5484E+02	-1.2031E+04
17. Raydan 1 Function (n = 1000) [2]	2.24 (1)	5.0050E+04	10.33	5.0050E+04	1.84E+02 (failed)	3.7229E+06 2.9262E+06	\ \
18. Raydan 2 Function (n = 1000) [2]	0.58 (1)	1.0000E+03	7.60	1.0000E+03	1.91E+02 (failed)	9.0831E+04 7.5209E+04	\ \
19. Extended Tridiagonal 1 Function(n = 1000) [2]	4.69 (1)	6.6911E-07	4.50 (failed)	2.9700E+02	2.82E+02 (failed)	3.2860E+06 3.0942E+06	\ \
20. Extended quadratic penalty QP1 Function (n = 1000) [2]	1.26E+02 (38)	3.9900E+03	3.30 (failed)	3.9962E+03	1.60E+02 (failed)	4.1622E+08 3.9182E+08	\ \
21. Extended quadratic penalty QP2 Function(n = 1000) [2]	6.5026 (3)	3.3900E-18	3.38 (failed)	1.0000E+04	1.97E+02 (failed)	4.1701E+08 3.9881E+08	\ \
22. Quadratic QF2 Function (n = 1000) [2]	9.54E+01 (46)	-1.0000	3.66	-1.0000	1.55E+02 (failed)	1.4025E+07 1.3692E+07	\ \
23. Extended PSC1 Function (n = 1000) [2]	3.38 (8)	3.8659E+02	3.3229 (failed)	5.0000E+02	1.62E+02	3.8659E+02 3.8659E+02	\ \
24. Extended BD1 Function (n = 1000) [2]	8.47E+01 (34)	0	2.50	0	1.86E+02 (failed)	5.2441E+06 4.8431E+06	\ \
25. Extended Cliff Function (n = 1000) [2]	3.59 (1)	9.9891E+01	5.44 (failed)	2.3779E+02	1.86E+02 (failed)	8.4363E+127 4.9166E+122	\ \
26. Perturbed quadratic diagonal Function (n = 1000) [2]	1.61 (1)	2.5157E-15	3.17	0	1.73E+02 (failed)	1.0437E+05 1.0292E+05	\ \
27. Extended Hiebert Function (n = 1000) [2]	6.13 (2)	6.2344E-19	2.21E+01 (failed)	1.2500E+12	1.71E+02 (failed)	1.2489E+12 1.2488E+12	\ \
28. Extended TET Function (n = 1000) [2]	1.71 (1)	1.2796E+03	3.65	1.2796E+03	2.12E+02 (failed)	7.2865E+15 4.9321E+15	\ \
29. Diagonal 1 Function (n = 1000) [2]	2.11 (1)	-2.7068E+06	3.54 (failed)	-2.2288E+06	1.99E+02 (failed)	6.5218E+05 5.9809E+05	\ \
30. Diagonal 3 Function (n = 1000) [2]	1.20E+01 (6)	-5.0046E+05	1.83E+01 (failed)	-4.8990E+05	2.36E+02 (failed)	5.3887E+04 4.1552E+04	\ \
31. Diagonal 5 Function (n = 1000) [2]	0.85 (1)	6.9314E+02	1.16E+01	6.9314E+02	2.97E+02 (failed)	3.6237E+03 3.5769E+03	\ \
32. Extended Maratos Function (n = 1000) [2]	4.39E+01 (3)	-5.0031E+02	3.63 (failed)	-5.0026E+02	1.66E+02 (failed)	1.8384E+08 1.7864E+08	\ \
33. EG2 Function (n = 1000) [2]	4.04 (5)	-9.9850E+02	2.71 (failed)	-9.2965E+02	2.15E+02 (failed)	-6.0856E+02 -6.1564E+02	\ \
34. SINQUAD Function (n = 1000) [2]	3.20 (1)	0	3.30	0	2.16E+02 (failed)	8.2895E+05 8.0685E+05	\ \

Table 2: Numerical results of small-scale unconstrained optimization problems.

Problem	CNMGE		GlobalSearch.m		DE.m		Global Minimum
	CPU (s) (Num. Loc. Sol.)	min $f(x)$	CPU (s)	min $f(x)$	CPU (s)	avg $f(x)$ min $f(x)$	
35. Griewank Function (n = 10) [44, 50]	5.51E-01 (215)	2.4924E-08	2.66	1.7187E-12	2.93	0 0	0
36. Levy Function N. 13 (n = 2) [50]	5.19E-01 (96)	1.5833E-14	2.85 (failed)	0.1098	2.81	1.3498E-31 1.3498E-31	0
37. Hosaki Function (n = 2) [3]	4.34E-01 (69)	-2.3458	2.62 (failed)	-1.1277	2.64 (failed)	0 0	-2.3458
38. Beale Function (n = 2) [44, 50]	7.16E-02 (37)	0	3.80	0	2.73	0 0	0
39. Easom Function (n = 2) [50]	4.08E-01 (22)	-1.0000	2.64	-1.0000	2.72	-0.3918 -1.0000	-1
40. Price Function (n = 2) [3]	3.14E-02 (94)	4.0665E-12	5.21	1.1149E-12	2.65	1.9420E-12 2.5781E-15	0
41. Branin Function (n = 2) [50]	7.00E-02 (37)	0.3978	2.21	0.3978	2.64	0.3978 0.3978	0.3978
42. Trecanni Function (n = 2) [3]	6.44E-02 (41)	3.9713E-18	2.43	1.8313E-15	2.74	-2.1316E-15 3.5527E-15	0
43. Booth Function (n = 2) [50]	8.79E-02 (1)	1.3603E-15	2.45	4.1193E-15	2.64	0 0	0
44. Matyas Function (n = 2) [50]	3.57E-01 (7)	0	2.11	6.94E-16	2.74	3.38E-66 1.1822E-68	0
45. McCormick Function (n = 2) [50]	3.57E-01 (73)	-1.9132	2.99	-1.9132	2.71	-1.9132 -1.9131	-1.9132
46. Power Sum Function (n = 4) [50]	6.87E-01 (5)	4.4121E-18	4.19	2.0472E-07	2.89 (failed)	1.23E-02 1.41E-03	0
47. Colville Function (n = 4) [50]	7.61E-02 (1)	3.9630E-19	3.80	5.3436E-13	2.69 (failed)	4.6297E-03 3.7651E-04	0
48. Schaffer Function N. 2 (n = 2) [50]	6.91E-01 (277)	3.3306E-16	4.17	0	2.81 (failed)	2.9251E-01 2.9251E-01	0
49. Bohachevsky Function (n = 2) [50]	4.57E-01 (75)	1.0991E-13	2.43	0	2.67	0 0	0
50. Three-Hump Camel Function (n = 2) [50]	7.65E-01 (40)	1.2994E-13	2.32	3.2640E-16	2.95	7.5997E-188 7.2500E-191	0
51. Six-Hump Camel Function (n = 2) [50]	3.74E-01 (60)	-1.0316	2.29	-1.0316	2.65	-1.0316 -1.0316	-1.0316
52. Drop-Wave Function (n = 2) [50]	6.30E-01 (265)	-1.0000	3.12	-1.0000	2.67	-1.0000 -1.0000	-1
53. Perm Function 0, d, β (n = 4) [50]	7.42E-01 (36)	4.9548E-21	5.79 (failed)	6.3025E+02	2.92 (failed)	8.5521E-01 1.3934E-01	0
54. Hartmann 3-D Function (n = 3) [50]	8.02E-01 (125)	-3.8627	2.77	-3.8627	2.66 (failed)	-1.0234E-306 -1.0234E-305	-3.8627
55. Trefethen 4 Function (n = 2) [3]	6.16E-01 (68)	-3.3069	5.91 (failed)	-2.8376	2.75	-3.3069 -3.3069	-3.3069
56. Zettl Function (n = 2) [3]	6.53E-02 (3)	-0.0037	2.36	-0.0037	2.70	-0.0037 -0.0037	-0.0037
57. Exp2 Function (n = 2) [3]	4.65E-01 (28)	3.2248E-13	2.81	4.8793E-14	2.84	1.2778 3.9188E-33	0
58. Hansen Function (n = 2) [3]	5.44E-01 (230)	-1.7654E+02	2.98	-1.7654E+02	2.70	-1.7654E+02 -1.7654E+02	-1.7654E+02
59. Shcaffer Function N. 4 (n = 2) [50]	3.47E-01 (205)	0.2925	4.60	0.2925	2.71	0.2925 0.2925	0.2925
60. Holder Table Function (n = 2) [50]	5.64E-01 (253)	-19.2085	3.36	-19.2085	2.73	-19.2085 -19.2085	-19.2085
61. Gramacy & Lee (2012) Function (n = 1) [50]	3.91E-01 (16)	-0.8690	5.49E-01	-0.8690	2.61	-0.8690 -0.8690	-0.8690
62. Eggholder Function (n = 2) [50]	4.75E-01 (13)	-1.4354E+03	3.53 (failed)	-9.3533E+02	2.74	-9.5964E+02 -9.5964E+02	-9.5964E+02
63. Michalewicz Function (n = 2) [50]	7.26E-01 (73)	-1.8013	4.32	-1.8013	2.80	-1.8013 -1.8013	-1.8013
64. Box-Betts Exponential Quadratic Function (n = 3) [3]	5.99E-01 (61)	4.5787E-19	2.76	6.8000E-17	2.89	6.8604E-202 1.8478E-213	0
65. Cross-in-Tray Function (n = 2) [50]	3.40E-01 (25)	-2.0626	3.87	-2.0626	2.73	-2.0626 -2.0626	-2.0626
66. Himmeblau Function (n = 2) [50]	1.52E-01 (6)	1.5938e-23	2.43	0	2.63	0 0	0
67. Forrester et al. (2008) Function (n = 2) [50]	1.85E-01 (144)	-6.0207	2.75	-6.0207	2.41	-6.0207 -6.0207	-6.0207
68. Goldstein-Price Function (n = 2) [50]	1.65E-01 (8)	3.0000	2.71	3.0000	2.49	3.0000 3.0000	3

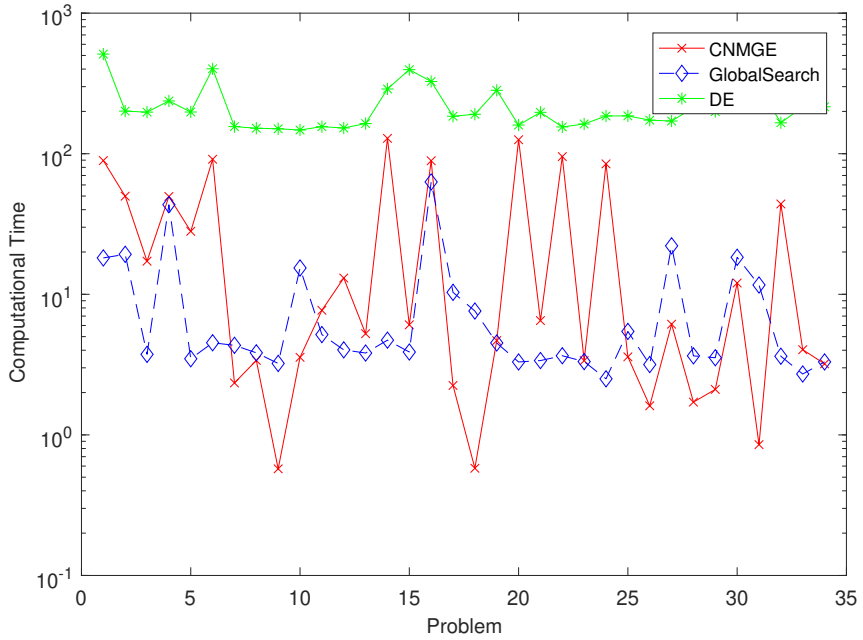

Fig. 1: The computational time of CNMGE, GlobalSearch and DE for the large-scale test problems.

6 Conclusions

Based on the deflation technique and the genetic evolution, we give an efficient continuation Newton for the unconstrained global optimization problem. The numerical results show that CNMGE (Algorithm 3) performs well to solve the large-scale optimization problems, especially the problems of which are difficult to be solved by the known global optimization methods. Therefore, CNMGE can be regarded as an alternative workhorse for the unconstrained global optimization problems.

Declarations

Funding: This work was supported in part by Grant 61876199 from National Natural Science Foundation of China, and Grant YJCB2011003HI from the Innovation Research Program of Huawei Technologies Co., Ltd..

Conflicts of interest/Competing interests: Not applicable.

Availability of data and material (data transparency): If it is requested, we will provide the test data.

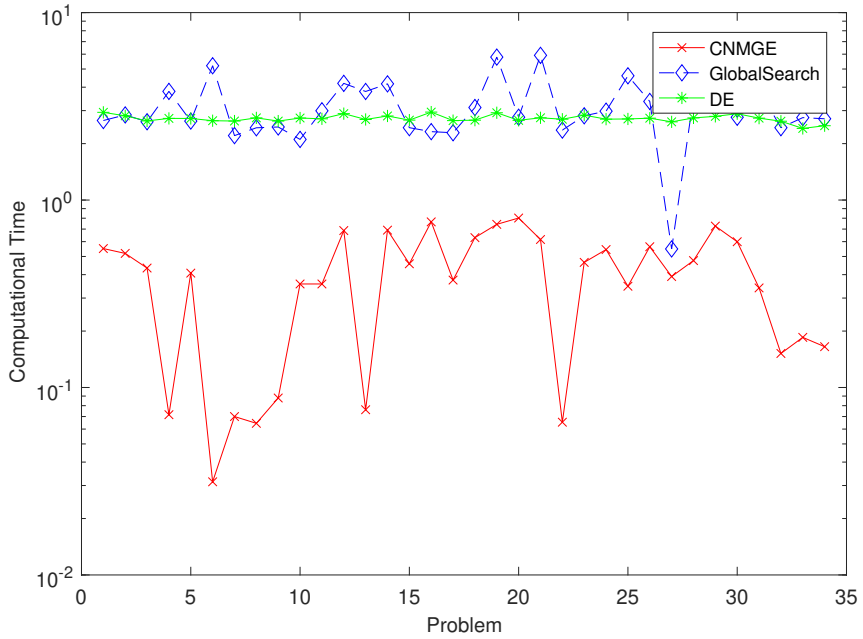

Fig. 2: The computational time of CNMGE, GlobalSearch and DE for the small-scale test problems.

Code availability (software application or custom code): If it is requested, we will provide the code.

References

1. Abbott, J. P.: *Numerical continuation methods for nonlinear equations and bifurcation problems*, Ph.D. Thesis, Computer Center, Australian National University, 1977.
2. Andrei, N.: *An unconstrained optimization test functions collection*, Advanced Modeling and Optimization **10** (2008), 147-161.
3. Adorio, E. P., Diliman, U. P.: *MVF-Multivariate test functions library in C for unconstrained global optimization*, available at http://www.geocities.ws/eadorio/mvf.pdf, 2005.
4. Allgower, E. L., Georg, K.: *Introduction to Numerical Continuation Methods*, SIAM, Philadelphia, PA, 2003.
5. Ascher, U. M., Petzold, L. R.: *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, Philadelphia, PA, 1998.
6. Axelsson, O., Sysala, S.: *Continuation Newton methods*, Comput. Math. Appl. **70** (2015), 2621-2637.
7. Boender, C. G. E.: *Bayestian stopping rules for multistart global optimization methods*, Math. Program. **37** (1987), 59-80.
8. Branin, F. H.: *Widely convergent method for finding multiple solutions of simultaneous nonlinear equations*, IBM J. Res. Dev. **16** (1972), 504-521.
9. Brown, K. M., Gearhart, W. B.: *Deflation techniques for the calculation of further solutions of a nonlinear system*, Numer. Math. **16** (1971), 334-342.
10. Conn, A. R., Gould, N., Toint, Ph. L.: *Trust-Region Methods*, SIAM, Philadelphia, PA, 2000.
11. Davidenko, D. F.: *On a new method of numerical solution of systems of nonlinear equations* (in Russian), Dokl. Akad. Nauk SSSR **88** (1953), 601-602.

12. Deuflhard, P.: *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*, Springer-Verlag, Berlin, 2004.
13. Deuflhard, P., Pesch, H. J., Rentrop, P.: *A modified continuation method for the numerical solution of nonlinear two-point boundary value problems by shooting techniques*, Numer. Math. **26** (1975), 327-343.
14. Dennis, J. E., Schnabel, R. B.: *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Philadelphia, PA, 1996.
15. Dong, H. C., Song, B. W., Dong, Z. M., Wang, P.: *Multi-start space reduction (MSSR) surrogate-based global optimization method*, Struct. Multidisc. Optim. **54** (2016), 907-926.
16. Elhara, O., Varelas, K., Nguyen, D., Tusar, T., Brockhoff, D., Hansen, N., Auger, A.: *COCO: The large scale black-box optimization benchmarking (bbob-largescale) Test Suite*, arXiv preprint available at https://arxiv.org/abs/1903.06396, 2019.
17. Gottlieb, S., Shu, C. W.: *Total variation diminishing Runge-Kutta schemes*, Math. Comput. **67**, 73-85 (1998).
18. Gottlieb, S., Shu, C. W., Tadmor, E.: *Strong stability preserving high order time discretization methods*, SIAM Rev. **43**, 89-112 (2001).
19. Golub, G. H., Van Loan, C. F.: *Matrix Computation* (4th ed.), The John Hopkins University Press, Baltimore, 2013.
20. Heris, M. K.: *Differential Evolution (DE) in MATLAB*, https://yarpiz.com/231/ypea107-differential-evolution, Yarpiz, 2015.
21. Hairer, E., Wanner, G.: *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems*, 2nd ed., Springer-Verlag, Berlin, 1996.
22. Hart, W. E.: *Adaptive Global Optimization with Local Search*, Ph.D. dissertation, University of California, San Diego, CA, USA, 1994.
23. Higham, D. J.: *Trust region algorithms and timestep selection*, SIAM J. Numer. Anal. **37** (1999), 194-210.
24. Kearfott, R. B.: *Rigorous Global Search: Continuous Problems*, Nonconvex Optimization and Applications, Kluwer Academic, Dordrecht, 1996.
25. Kelley, C. T.: *Solving Nonlinear Equations with Newton's Method*, SIAM, Philadelphia, PA, 2003.
26. Kelley, C. T.: *Numerical methods for nonlinear equations*, Acta Numer. **27** (2018), 207-287.
27. Lavor, C., Maculan, N.: *A function to test methods applied to global minimization of potential energy of molecules*, Numer. Algorithms **35** (2004), 287-300.
28. Leung, Y.-W., Wang, Y. P.: *An orthogonal genetic algorithm with quantization for global numerical optimization*, IEEE Trans. Evol. Comput. **5** (2001), 41-53.
29. Liu, S.-T., Luo, X.-L.: *A method based on Rayleigh quotient gradient flow for extreme and interior eigenvalue problems*, Linear Algebra Appl. **432** (2010), 1851-1863.
30. Luo, X.-L.: *Singly diagonally implicit Runge-Kutta methods combining line search techniques for unconstrained optimization*, J. Comput. Math. **23**, 153-164 (2005)
31. Luo, X.-L., Kelley, C. T., Liao, L.-Z., Tam, H.-W.: *Combining trust region techniques and Rosenbrock methods to compute stationary points*, J. Optim. Theory Appl. **140** (2009), 265-286.
32. Luo, X.-L.: *A second-order pseudo-transient method for steady-state problems*, Appl. Math. Comput., **216** (2010), 1752-1762.
33. Luo, X.-L.: *A dynamical method of DAEs for the smallest eigenvalue problem*, J. Comput. Sci. **3** (2012), 113-119.
34. Luo, X.-L., Lv, J.-H., Sun, G.: *Continuation method with the trusty time-stepping scheme for linearly constrained optimization with noisy data*, Optim. Eng., published online at published online, http://doi.org/10.1007/s11081-020-09590-z, 1-32, January 2021.
35. Luo, X.-L., Xiao, H., Lv, J.-H.: *Continuation Newton methods with the residual trust-region time-stepping scheme for nonlinear equations*, Numer. Algorithms, published online at http://doi.org/10.1007/s11075-021-01112-x, 1-25, April 2, 2021.
36. Luo, X.-L., Yao, Y. Y.: *Primal-dual path-following methods and the trust-region strategy for linear programming with noisy data*, J. Comput. Math., published online at http://doi.org/10.4208/jcm.2101-m2020-0173, or available at http://arxiv.org/abs/2006.07568, 1-21, March 15, 2021.
37. Luo, X.-L., Xiao, H., Lv, J.-H., Zhang, S.: *Explicit pseudo-transient continuation and the trust-region updating strategy for unconstrained optimization*, Appl. Numer. Math., **165** (2021), 290-302.
38. Luo, X.-L., Xiao, H.: *Generalized continuation Newton methods and the trust-region updating strategy for the underdetermined system*, J. Sci. comput. **88** (2021), published online at http://doi.org/10.1007/s10915-021-01566-0, 1-22.

39. Luo, X.-L., Xiao, H.: *The regularization continuation method with an adaptive time step control for linearly equality-constrained optimization problems*, submitted, arXiv preprint available at http://arxiv.org/abs/2106.01122, July 7, 2021.
40. Man, K. F., Tang, K. S., Kwong, S.: *Genetic Algorithms: Concepts and Designs*, Springer, Berlin, 1999.
41. Macêdo, M. J. F. G., Karas, E. W., Costa, M. F. P., Rocha, A. M. A. C.: *Filter-based stochastic algorithm for global optimization*, J. Glob. Optim. **77** (2020), 777-805.
42. MATLAB v9.8.0 (R2020a), The MathWorks Inc., http://www.mathworks.com, 2020.
43. Mitchell, M.: *An Introduction to Genetic Algorithms*, MIT press, Cambridge, MA, 1996.
44. Moré, J. J., Garbow, B. S., Hillstrom, K. E.: *Testing unconstrained optimization software*, ACM Trans. Math. Soft. **7** (1981), 17-41.
45. Moscato, P.: *On evolution, search, optimization, gas and martial arts: Toward memetic algorithms*, Technical report C3P, Vol. 826, California Institute of Technology, Caltech Concurrent Computing Program, Pasadena, California, 1989.
46. Nocedal, J., Wright, S. J.: *Numerical Optimization*, Springer-Verlag, Berlin, 1999.
47. Ortega, J. M., Rheinboldt, W. C.: *Iteration Solution of Nonlinear Equations in Several Variables*, SIAM, Philadelphia, PA, 2000.
48. Regis, R. G., Shoemaker, C. A.: *A quasi-multistart framework for global optimization of expensive functions using response surface models*, J. Glob. Optim. **56** (2013), 1719-1753.
49. Shampine, L. F., Gladwell, I., Thompson, S.: *Solving ODEs with MATLAB*, Cambridge University Press, Cambridge, 2003.
50. Surjanovic, S., Bingham, D.: *Virtual library of simulation experiments: test functions and datasets*, available at http://www.sfu.ca/~ssurjano, January 2020.
51. Sun, J., Garibaldi, J. M., Krasnogor, N., Zhang, Q.: *An intelligent muti-restart memetic algorithm for box constrained global optimisation*, Evol. Comput. **21** (2014), 107-147.
52. Sergeyev, Y. D., Kvasov, D. E., Mukhametzhanov, M. S.: *On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget*, Sci. Rep. **8** (2018), 1-9.
53. Sun, W.-Y, Yuan, Y.-X.: *Optimization Theory and Methods: Nonlinear Programming*, Springer, Berlin, 2006.
54. Tanabe, K.: *Continuous Newton-Raphson method for solving an underdetermined system of nonlinear equations*, Nonlinear Anal. **3** (1979), 495-503.
55. Teughels, A., Roeck, G. De, Suykens, J. A. K.: *Global optimization by coupled local minimizers and its application to FE model updating*, Comput. Sturct. **81** (2003), 2337-2351.
56. Ugray, Z., Lasdon, L., Plummer, J., Glover, F., Kelly, J., Marti, R.: *Scatter search and local NLP solvers: A multistart framework for global optimization*, INFORMS J. Comput. **19** (2007), 328-340.
57. Yarpiz, *Differential Evolution (DE)* https://www.mathworks.com/matlabcentral/fileexchange/52897-differential-evolution-de, MATLAB Central File Exchange, 2021.
58. Yuan, Y.-X.: *Recent advances in trust region algorithms*, Math. Program. **151** (2015), 249-281.