Neurocomputing 273 (2018) 237–250

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A dynamic state transition algorithm with application to sensor network localization

Xiaojun Zhou[a], Peng Shi[b], Cheng-Chew Lim[b], Chunhua Yang[a,*], Weihua Gui[a]

[a] School of Information Science and Engineering, Central South University, Changsha 410083, China
[b] School of Electrical and Electronic Engineering, The University of Adelaide, Adelaide, SA 5005, Australia

ARTICLE INFO

Article history:
Received 8 April 2017
Revised 22 July 2017
Accepted 8 August 2017
Available online 31 August 2017

Communicated by Dr. Ma Lifeng Ma

Keywords:
State transition algorithm
Dynamic adjustment
Sensor network localization
Global optimization

ABSTRACT

The sensor network localization (SNL) problem aims to reconstruct the positions of all the sensors in a network with given distance between pairs of sensors and within the radio range between them. It is proved that the computational complexity of the SNL problem is NP-hard, and semi-definite programming or second-order cone programming relaxation methods can only solve some special problems of this kind. In this study, a stochastic intelligent optimization method based on the state transition algorithm is introduced to solve the SNL problem without additional assumptions and conditions on the problem structure. To transcend local optimality, a novel dynamic adjustment strategy called "risk and restoration in probability"is incorporated into the state transition algorithm. An empirical study is investigated to appropriately choose the risk probability and restoration probability, yielding the dynamic state transition algorithm, which is further improved with gradient-based refinement. The refined dynamic state transition algorithm is applied to the SNL problem, and satisfactory simulation results show the effectiveness of the proposed approach.

1. Introduction

In recent decades, ad hoc wireless sensor networks have received considerable attention due to easy installation and simple operation [1–9]. A typical sensor network consists of a large set of sensors distributed in a geographical area. Sensor nodes collect the local information, such as temperature, humidity, and vibration motion, and communicate with other neighboring nodes, which are the nodes if the distance between them is below certain radio range. The sensor data collected from these nodes are relevant only if the node positions are known. Although locating these positions can be achieved through manual configuration or by using Global Positioning System (GPS) techniques, neither works well and both have physical limitations. As a result, techniques to estimate node positions have shifted to develop methods that rely only on the distance measures between neighboring nodes. The distance information could be based on criteria such as time of arrival, time-difference of arrival, or received RF (radio frequency) signal strength. We may further assume that we already know the exact positions of a few sensor nodes, which we will refer to as anchors, and it should be noted that the determination of anchor points is costly in real-world applications. Given the positions of the anchor nodes and the relative distance between neighboring nodes, the problem of finding the positions of all the unknown sensor nodes is called the sensor network localization (SNL) problem. The SNL problem is a nonlinear equation problem in its original form, whereas by using the least squares method, the SNL problem can be converted into a non-convex optimization problem [10].

The difficulty of locating the unknown sensors accurately is three-fold: (1) the distance measurements may contain some noise or uncertainty, (2) it is not easy to identify the sufficient conditions for the sensor network to be localizable, and (3) the sensor network localization problem is proved to be NP-hard [11]. The NP-hardness has led to efforts being directed at solving this problem approximately or solving it completely under certain conditions. Semi-definite programming (SDP) relaxation has been widely used for the SNL problem [10,12–14], but the solutions obtained by SDP relaxation are not generally optimal or are even infeasible. It is therefore necessary to round the SDP solution to a suboptimal and feasible one. Since the distance measurements inevitably contain noise or uncertainties, methods for rounding the SDP solution may become not so robust and reliable. Second-order cone programming (SOCP) relaxation has also found applications for the

* Corresponding author.
E-mail address: ychh@csu.edu.cn (C. Yang).

http://dx.doi.org/10.1016/j.neucom.2017.08.010
0925-2312/

SNL problem [15,16]. It is shown that even if it is weaker than SDP relaxation, the SOCP relaxation has a simpler structure and nicer properties that can make it useful as a problem preprocessor due to its faster speed. It should be noted that both SDP and SOCP relaxation in essence are gradient-based methods, and other gradient-based methods for the SNL problem can also be found in [17–21].

To the best of our knowledge, there exist very few stochastic methods for the SNL problem. While some particle swarm optimization algorithms have been used to solve the SNL problem [22–24], but the sizes of their problems in practice were limited to less than 100. The state transition algorithm (STA) has emerged in recent years as a novel stochastic intelligent optimization method. In the STA, a solution to an optimization problem is regarded as a state and a transformation of current solution is regarded as a state transition [25]. It uses state space representation, thus it can generate candidate solutions in a unified framework, and the execution operators for generating candidate solutions are expressed as state transition matrices, which makes it easy to understand and flexible to implement. In the continuous STA, there exist four state transformation operators: rotation, translation, expansion, and axesion. These operators have special characteristics that cover the local and global search capability. For example, the rotation operator can search in a hypersphere within a given radius, thus belonging to a local search, while the expansion operator can explore the whole space, thus belonging to a global search. The job specialization is convenient for users when manipulating the STA according to their demands. The strong global search ability and adaptability of the STA have been demonstrated by comparison with other global optimization algorithms and real-world applications [25–32].

In this paper, we use the STA to find a solution for the SNL problem with a size that can exceed 100. The STA is a stochastic global optimization algorithm, thus it can find a global solution to the SNL problem without additional assumptions and conditions. Nevertheless, like most stochastic optimization algorithms, it will inevitably get trapped in local minima with a limited amount of time. To transcend local optimality, a dynamic adjustment strategy called "risk and restoration in probability" is proposed to improve the global search ability. Risk in probability means that a relatively worse solution is accepted for the next iteration with probability p_{risk}, while restoration in probability means that the historical best solution is restored with probability p_{rest}. The values of these two probability are investigated by an empirical study in this study. With the proposed dynamic adjustment strategy, a dynamic STA with refinement is presented for the SNL problem. Several simulation results have demonstrated the effectiveness of the proposed approach.

The main contribution of this paper is three-fold. First, a fast rotation transformation operator is designed to reduce the computational complexity. Second, a dynamic STA with "risk and restoration in probability" strategy is proposed to escape from local optimality, and a good combination of the risk probability and the restoration probability is obtained by an empirical study. Third, the proposed dynamic STA with refinement is successfully applied to the sensor network localization problem.

The rest of this paper is organized as follows. The sensor network localization problem is formulated in Section 2. Then, in Section 3, we give a brief review of the basic STA. In Section 4, a fast rotation transformation is proposed, the local convergence analysis of the basic STA is discussed and a dynamic adjustment strategy is proposed to improve its global search ability. The proposed dynamic STA with refinement is applied to the sensor network localization problem in Section 5, and the concluding remarks are given in Section 6.

2. Sensor network localization problem

Given m anchor points $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m \in \mathbb{R}^d$ (d is usually 2 or 3, $d = 2$ in this study), the Euclidean distance d_{ij} between the ith and jth anchor points for $(i, j) \in N_x$, and the distance $\bar{d}_{ik}$ between the ith sensor and kth anchor points for $(i, k) \in N_a$, with $N_x = \{(i, j) : \|\boldsymbol{x}_i - \boldsymbol{x}_j\| = d_{ij} \le r_d\}$ and $N_a = \{(i, k) : \|\boldsymbol{x}_i - \boldsymbol{a}_k\| = \bar{d}_{ik} \le r_d\}$, where, r_d is the radio range, the sensor network localization (SNL) problem is to find n distinct sensor points $\boldsymbol{x}_i, i = 1, \ldots, n$, such that

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 = d_{ij}^2, \forall (i, j) \in N_x, \tag{1a}$$

$$\|\boldsymbol{x}_i - \boldsymbol{a}_k\|^2 = \bar{d}_{ik}^2, \forall (i, k) \in N_a. \tag{1b}$$

Two most commonly used methods to solve the SNL problem are SDP relaxation and SOCP relaxation. Let $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$ be the unknown matrix to be determined and $Y = X^T X$. Then Eq. (1a) can be rewritten as

$$e_{ij}^T Y e_{ij} = d_{ij}^2, \forall (i, j) \in N_x$$
$$Y = X^T X,$$

where $e_{ij} \in \mathbb{R}^n$ is the sparse vector with nonzero values 1 and -1 at the ith and the jth positions, respectively. The trick of SDP relaxation method is to relax $Y = X^T X$ to $Y \succeq X^T X$ and then reformulate it to a SDP problem. The similarity applies for the SOCP relaxation method. Firstly, Eq. (1a) can be equivalently written as

$$\min \sum_{(i,j)\in N_x} |y_{ij} - d_{ij}^2|$$
$$\text{s.t.} \quad y_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2, \forall (i, j) \in N_x.$$

By introducing auxiliary variables t_{ij}, relaxing $y_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$ to $y_{ij} \ge \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$ yields the following SOCP problem

$$\min \sum_{(i,j)\in N_x} t_{ij}$$
$$\text{s.t.} \quad y_{ij} \ge \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2, \forall (i, j) \in N_x$$
$$t_{ij} \ge |y_{ij} - d_{ij}^2|, \forall (i, j) \in N_x.$$

The aforementioned relaxation methods have an inevitable limitation, that is, the original problem is equivalent to the relaxation problem only if these inequality constraints are active. If they are not, there exists an inherent relaxation gap, making the solution of the relaxation problem infeasible for the original problem. In other words, these relaxation methods only work when a SNL problem has certain special structures. To be more practical, a general SNL problem is considered to be solved in this study. Since distances d_{ij} and $\bar{d}_{ik}$ may contain noise, making Eqs. (1a) and (1b) infeasible, we reformulate the SNL problem using the least squares method as the following nonconvex optimization problem

$$\min \sum_{(i,j)\in N_x} (\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 - d_{ij}^2)^2 + \sum_{(i,k)\in N_a} (\|\boldsymbol{x}_i - \boldsymbol{a}_k\|^2 - \bar{d}_{ik}^2)^2. \tag{2}$$

Considering that local minima exist in the nonconvex optimization problem, the goal of this study is to find a global solution to this problem using a novel stochastic global optimization method called STA. To expedite the search process and to improve its global search ability, some modifications are added to the basic STA. In the following section, a brief review of STA is presented.

3. A brief review of the basic STA

In numerical optimization, it usually adopts an iterative method to transform the current solution $\boldsymbol{x}_k$ to the next one $\boldsymbol{x}_{k+1}$ via designing appropriate operators. For deterministic optimization methods, like gradient descent method, the first order differential

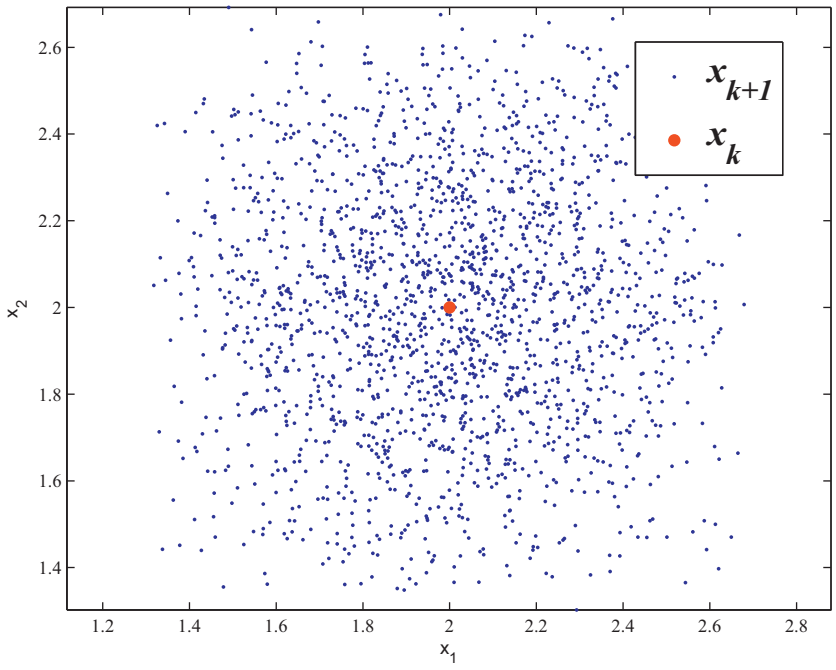

Fig. 1. Illustration of the rotation transformation ($\alpha = 1$). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article).

operator is added to generate the next candidate solution, while in Newton's method, both the first and second order differential operators are utilized. For stochastic optimization methods, like crossover and mutation operators in genetic algorithm, and position and velocity update operators in particle swarm optimization, they all aim to transform current solutions to next candidate solutions. Similarly, in a state transition way, a solution can be regarded as a state, and the transformation of a solution can be considered as a state transition process. On the basis of state space representation, the unified form of generating new candidate states in STA can be described as follows:

$$\begin{cases} \boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k \\ y_{k+1} = f(\boldsymbol{x}_{k+1}), \end{cases} \tag{3}$$

where $\boldsymbol{x}_k = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^n$ stands for a state corresponding to a solution of the optimization problem, $\boldsymbol{u}_k$ is a function of $\boldsymbol{x}_k$ and historical states, y_k is the fitness value at $\boldsymbol{x}_k$, A_k and B_k are state transition matrices, which are usually some transformation operators, and f is the objective function or evaluation function.

3.1. State transformation operators

Using space transformation, the following four special state transformation operators are designed to generate new candidate solutions.

(1) Rotation transformation (RT)

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha \frac{1}{n\|\boldsymbol{x}_k\|_2} R_r \boldsymbol{x}_k, \tag{4}$$

where α is a positive constant, called the rotation factor, $R_r \in \mathbb{R}^{n\times n}$, is a random matrix with its entries being uniformly distributed random variables defined on the interval $[-1, 1]$, and $\|\cdot\|_2$ is the 2-norm of a vector. This rotation transformation has the function of searching in a hypersphere with a maximal radius α,

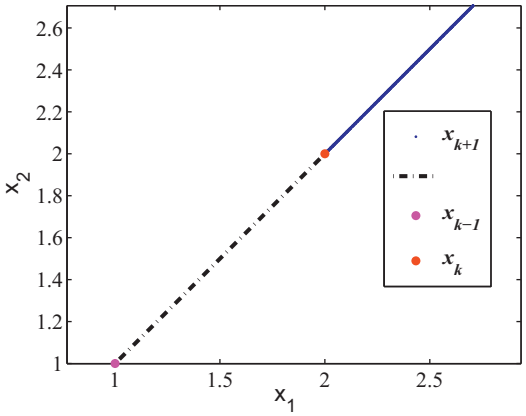

Fig. 2. Illustration of the translation transformation ($\beta = 1$).

as shown below

$$\begin{aligned} \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2 &= \|\alpha \frac{1}{n\|\boldsymbol{x}_k\|_2} R_r \boldsymbol{x}_k\|_2 \\ &= \frac{\alpha}{n\|\boldsymbol{x}_k\|_2} \|R_r \boldsymbol{x}_k\|_2 \\ &\le \frac{\alpha}{n\|\boldsymbol{x}_k\|_2} \|R_r\|_{m_\infty} \|\boldsymbol{x}_k\|_2 \le \alpha. \end{aligned} \tag{5}$$

The illustration of the RT in 2-D is given in Fig. 1, here, the current solution $\boldsymbol{x}_k = [x_1, x_2]^T = (2, 2)$, marked in red, and the next candidate solutions $\boldsymbol{x}_{k+1}$ are marked in blue by performing thousands of times of rotation transformation.

(2) Translation transformation (TT)

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \beta R_t \frac{\boldsymbol{x}_k - \boldsymbol{x}_{k-1}}{\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_2}, \tag{6}$$

where β is a positive constant, called the translation factor, and $R_t \in \mathbb{R}$ is a uniformly distributed random variable defined on the interval [0,1]. Fig. 2 shows that the translation transformation has

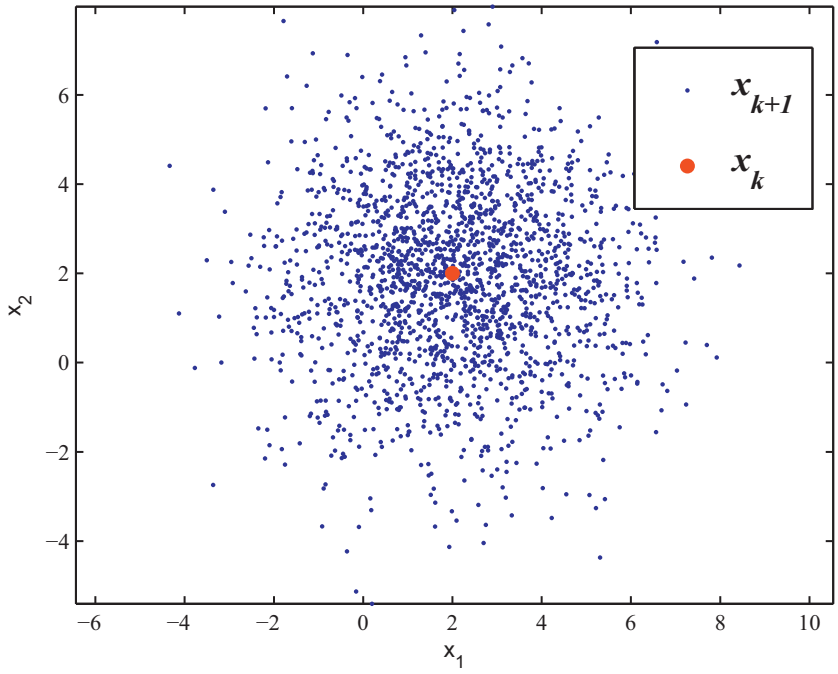

Fig. 3. Illustration of the expansion transformation ($\gamma = 1$).

the function of searching along a line from $\boldsymbol{x}_{k-1} = (1, 1)$ to $\boldsymbol{x}_k = (2, 2)$ at the starting point $\boldsymbol{x}_k$ with maximal length β.

(3) Expansion transformation (ET)

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \gamma R_e \boldsymbol{x}_k, \tag{7}$$

where γ is a positive constant, called the expansion factor, and $R_e \in \mathbb{R}^{n\times n}$ is a random diagonal matrix with its entries obeying the Gaussian distribution. Fig. 3 shows that the expansion transformation has the function of expanding the entries in $\boldsymbol{x}_k$ to the range of $[-\infty, +\infty]$, searching in the whole space. Here, $\boldsymbol{x}_k = (2, 2)$.

(4) Axesion transformation (AT)

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \delta R_a \boldsymbol{x}_k, \tag{8}$$

where δ is a positive constant, called the axesion factor, and $R_a \in \mathbb{R}^{n\times n}$ is a random diagonal matrix with its entries obeying the Gaussian distribution and with only one random position having nonzero value. As illustrated in Fig. 4, the axesion transformation is aiming to search along the axes. Here, $\boldsymbol{x}_k = (2, 2)$.

Remark 1. In the basic STA, from their intrinsic properties, the rotation transformation is used for exploitation (local search), the expansion is used for exploration (global search), the translation is a line search that is applied only when a better solution is found, and the axesion is for strengthening the single dimensional search.

3.2. Regular neighborhood and sampling

For a given solution, a candidate solution is generated by using one time of the aforementioned state transformation operators. Since the state transition matrix in each state transformation is random, the generated candidate solution is not unique. Based on the same given point, it is not difficult to imagine that a regular neighborhood will be automatically formed when using certain state transition operator, as illustrated from Figs. 1 to 4. In theory, the number of candidate solutions in the neighborhood is infinity; as a result, it is impractical to enumerate all possible candidate solutions.

Since the entries in state transition matrix obey certain stochastic distribution, for any given solution, the new candidate becomes a random vector and its corresponding solution (the value of a random vector) can be regarded as a sample. Considering that any two random state transition matrices in each state transformation are independent, several times of state transformation (called the degree of search enforcement, *SE* for short) based on the same given solution are performed for certain state transition operator, consisting of *SE* samples. It is not difficult to find that all of the *SE* samples are independent, and they are representatives of the neighborhood. Taking the rotation transformation for example, a total number of *SE* samples are generated in pseudocode as follows:

1: **for** $i \leftarrow 1$, *SE* **do**
2: $\quad \text{State}(:, i) \leftarrow \text{Best} + \alpha \frac{1}{n\|\text{Best}\|_2} R_r \text{Best}$
3: **end for**

where *Best* is the incumbent best solution, and *SE* samples are stored in the matrix *State*.

3.3. An update strategy

As mentioned above, based on the incumbent best solution, a total number of *SE* candidate solutions are sampled. A new best solution is selected from the candidate set by virtue of the evaluation function, denoted as *newBest*. Then, an update strategy based on greedy criterion is used to update the incumbent best as shown below

$$\text{Best} = \begin{cases} \text{newBest}, & \text{if } f(\text{newBest}) < f(\text{Best}), \\ \text{Best}, & \text{otherwise}. \end{cases}$$

3.4. Algorithm procedure of the basic STA

With the state transformation operators, sampling technique and update strategy, the basic STA can be described by the following pseudocode

As for detailed explanations, rotation(·) in above pseudocode is given for illustration purposes as follows

Remark 2. In the aforementioned pseudocodes, expansion(·), rotation(·) and axesion(·) are implementations of corresponding

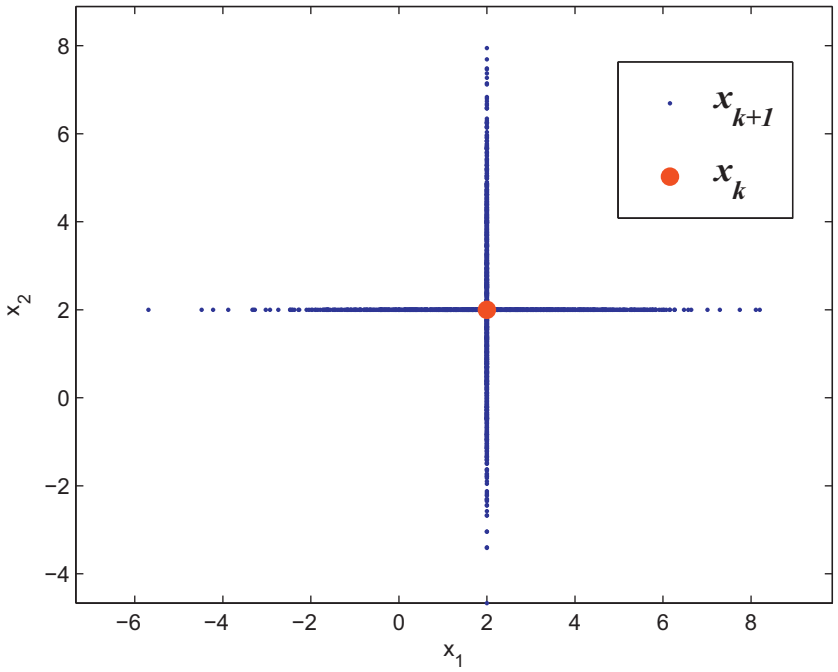

Fig. 4. Illustration of the axesion transformation ($\delta = 1$).

Algorithm 1 The basic STA.

```
procedure STA(funfcn, Best0, SE, α, β, γ, δ)
    Best ← Best0
    repeat
        if α < α_min then
            α ← α_max
        end if
        Best ← expansion(funfcn,Best,SE,β,γ)
        Best ← rotation(funfcn,Best,SE,α,β)
        Best ← axesion(funfcn,Best,SE,β,δ)
        α ← α/fc
    until the specified termination criterion is met
end procedure
```

state transformation, sampling technique and update strategy. funfcn is the name of a function that needs to be optimized. The rotation factor α decreases periodically from a maximum value $\alpha_{\max}$ to a minimum value $\alpha_{\min}$ exponentially with base fc, which is called lessening coefficient. op_rotate($\cdot$) and op_translate($\cdot$) represent the implementations of proposed sampling technique for rotation and translation operators, respectively, and fitness($\cdot$) represents the implementation of selecting the new best solution from SE samples. It should be noted that the translation operator is only executed when a solution better than the incumbent best solution can be found in the SE samples from rotation, expansion or axesion transformation. In the basic STA, the parameter settings are given as follows: $\alpha_{\max} = 1, \alpha_{\min} = 1e-4,\ \beta = 1, \gamma = 1, \delta = 1$, $SE = 30, fc = 2$.

4. Dynamic STA

Although the basic STA performs well for the majority of the benchmark functions as shown in the experimental results of [25], it behaves weakly for the Rosenbrock function. In fact, the Rosenbrock function is considered as a hard case for most deterministic and stochastic optimization algorithms, and more specially, its structure, known as the fourth-order polynomial, is similar to the SNL problem in our study. Furthermore, from the experimental results in [28], as the iterations increase, the objective value of Rosenbrock function still decreases, but at a very slow speed. That is to say, the global search ability of STA is not bad but its local search for the fourth-order polynomial (the Rosenbrock problem and the SNL problem belong to this type) is not so good. As a result, in this section, firstly, a fast rotation transformation is proposed to replace the original one. Then, a convergence analysis of STA is discussed based on monotone convergence theorem. Finally, we focus on how to jump out from local minima using a dynamic adjustment strategy, called "risk and restoration in probability", in other words, the dynamic STA risks accepting a relatively worse solution with one probability and restores the historical best solution with another probability.

```
function ROTATION(funfcn,Best,SE,α, β)
    oldBest ← Best
    fBest ← feval(funfcn,oldBest)
    State ← op_rotate(Best,SE,α)
    [newBest,fnewBest] ← fitness(funfcn,State)
    if fnewBest < fBest then
        Best ← newBest
        fBest ← fnewBest
        State ← op_translate(oldBest,newBest,SE,β)
        [newBest,fnewBest] ← fitness(funfcn,State)
        if fnewBest < fBest then
            fBest ← fnewBest
            Best ← newBest
        end if
    end if
end function
```

4.1. Fast rotation transformation

As illustrated in Fig. 1, the rotation transformation searches a hypersphere within a given radius, but a random matrix needs to be generated for every rotation transformation. To reduce the computational complexity, a fast rotation transformation is proposed as follows:

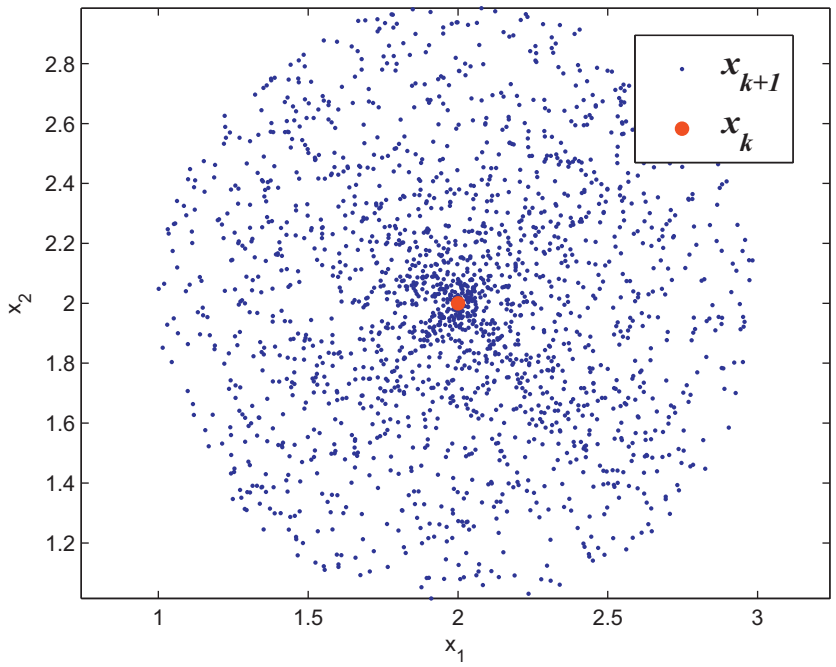

Fig. 5. Illustration of the fast rotation transformation.

(5) Fast rotation transformation

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha \hat{R}_r \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|_2}, \tag{9}$$

where $\hat{R}_r \in \mathbb{R}$ is a uniformly distributed random variable defined on the interval $[-1,1]$ and $\boldsymbol{u} \in \mathbb{R}^n$ is a vector with its entries being uniformly distributed random variables defined on the interval $[-1,1]$. It is easy to verify that

$$\begin{aligned} \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2 &= \left\| \alpha \hat{R}_r \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|_2} \right\|_2 \\ &= \alpha \|\hat{R}_r\|_2 \le \alpha. \end{aligned} \tag{10}$$

The illustration of the fast rotation transformation, where $\boldsymbol{x}_k = (2, 2)$, is given in Fig. 5. Compared with the original rotation operator, the fast rotation transformation has low computational complexity since the new random variable $\hat{R}_r$ is a scalar and not a matrix R_r. The computational time for two rotation operators is compared in Fig. 6, where, at each fixed dimension, we run the two rotation operators for 10,000 times on an Intel(R) Core(TM) i3-2310M CPU @2.10GHz under Windows 7. It is clear that the fast rotation operator can save a significant amount of computing time.

4.2. Convergence analysis of STA

In this subsection, we show that the basic STA can at least converge to a local minimum. From a control theory perspective, the evolution of the incumbent best state $\boldsymbol{x}_k^*$ in STA using the greedy criterion can be regarded as a discrete-time switched system

$$\boldsymbol{x}_{k+1}^* = \begin{cases} A_k \boldsymbol{x}_k^* + B_k \boldsymbol{u}_k, \text{ if } f(A_k \boldsymbol{x}_k^* + B_k \boldsymbol{u}_k) < f(\boldsymbol{x}_k^*), \\ \boldsymbol{x}_k^*, \quad \text{otherwise.} \end{cases} \tag{11}$$

A switched system contains some interesting stability phenomena. For instance, even when all the subsystems are exponentially stable, a switched system may have divergent trajectories for certain switching signals. That is to say, the stability of a switched system depends not only on the dynamics of each subsystem but also on the properties of switching signals.

Theorem 1. *The sequences* $\{f(\boldsymbol{x}_k^*)\}_{k=0}^{\infty}$ *generated by the STA can at least converge to a local minimum,* i.e.,

$$\lim_{k \to \infty} f(\boldsymbol{x}_k^*) = f(\bar{\boldsymbol{x}}^*) \tag{12}$$

where $\bar{\boldsymbol{x}}^*$ *is a local minimum.*

Proof. Since $f(\boldsymbol{x}_{k+1}^*) \le f(\boldsymbol{x}_k^*)$ and $f(\boldsymbol{x}_k^*)$ is bounded below on $\mathbb{R}^n$, *i.e.*, the global solution can be achieved, the sequence $\{f(\boldsymbol{x}_k^*)\}_{k=0}^{\infty}$ converges according to the *monotone convergence theorem*.

Denote $\bar{\boldsymbol{x}}^*$ as the limiting point of the sequence $\{f(\boldsymbol{x}_k^*)\}_{k=0}^{\infty}$. Due to the rotation operator and accepting criteria used in the STA, it is easy to find that

$$f(\bar{\boldsymbol{x}}^*) \le f(\boldsymbol{x}), \quad \forall \, \|\boldsymbol{x} - \bar{\boldsymbol{x}}^*\| \le \alpha_{\min} \tag{13}$$

that is to say, $\bar{\boldsymbol{x}}^*$ is a local minimum when the value of $\alpha_{\min}$ is sufficiently small. This completes the proof. □

4.3. Transcending local optimality

Note that although the sequence $\{f(\boldsymbol{x}_k^*)\}_{k=0}^{\infty}$ converges when k approaches infinity, it does not mean that the sequence converges to the global solution. This phenomenon is called "premature convergence" in the evolutionary computation community. From a practical point of view, "premature convergence" is inevitable since there is no prior knowledge to judge whether a solution is indeed the global solution. Thus, how to jump out of local minima becomes a significant issue. Risking a relatively bad solution in probability is an effective strategy to escape from local minima, as indicated in simulated annealing [33], but it is computationally expensive to achieve convergence [34]. In this study, a new strategy called "risk and restoration in probability" is proposed. For each state transformation in the inner loop, a relatively worse solution is accepted based on the risk probability. While in the outer loop, the best solution, which is stored in external archive,

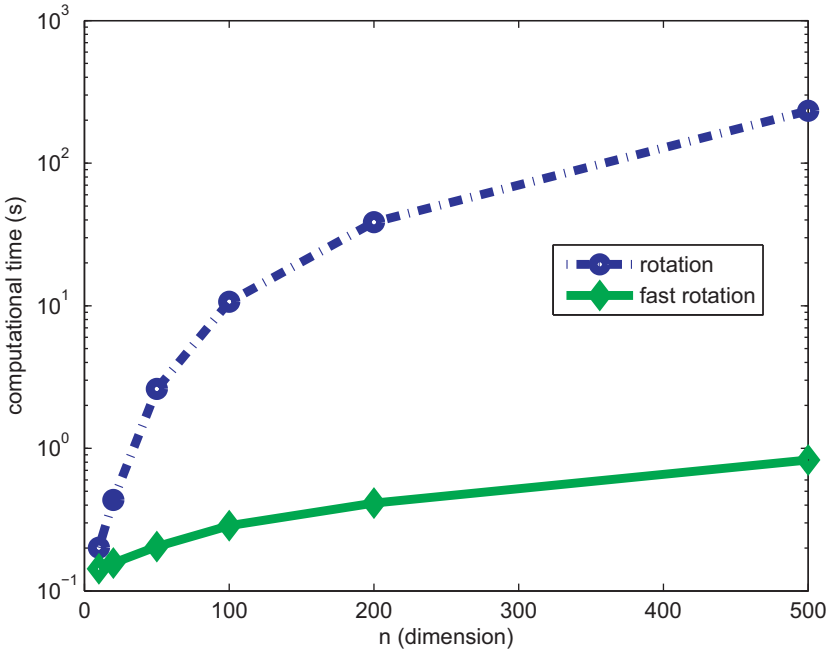

Fig. 6. Comparison of the computational time for rotation operators.

is restored to update the incumbent current solution. The pseudocode for the proposed dynamic STA in the outer loop is given below

Algorithm 2 The dynamic STA.

procedure STA(funfcn, Best0, SE, $\alpha, \beta, \gamma, \delta$)
  repeat
    if $\alpha(\beta,\gamma,\delta) < \alpha_{\min}(\beta_{\min},\gamma_{\min},\delta_{\min})$ **then**
      $\alpha(\beta,\gamma,\delta) \leftarrow \alpha_{\max}(\beta_{\max},\gamma_{\max},\delta_{\max})$
    end if
    [Best,fBest] ← expansion(funfcn,Best,SE,β,γ)
    [Best,fBest] ← rotation_fast(funfcn,Best,SE,α,β)
    [Best,fBest] ← axesion(funfcn,Best,SE,β,δ)
    if fBest < fBest* **then**
      Best* ← Best
      fBest* ← fBest
    end if
    if *rand* < p_{rest} **then** ▷ restoration in probability
      Best ← Best*
      fBest ← fBest*
    end if
    $\alpha(\beta,\gamma,\delta) \leftarrow \frac{\alpha(\beta,\gamma,\delta)}{fc}$
  until the maximum number of iterations is met
end procedure

In the inner loop, taking the expansion function for example, the pseudocode is given below

Remark 3. In the dynamic STA, a fast rotation operator is introduced. Furthermore, not only α, but also β, γ and δ decrease from a maximum value to a minimum value exponentially with base *fc*, which will be helpful for exploitation in the later stage. Best and fBest are the incumbent current solution and its function value, Best* and fBest* are the best solution and its function value in history, respectively, and they are all kept in an external archive. p_{rest} and p_{risk} are the restoration probability and the risk probability, respectively. Compared with the accepting criteria in simulated annealing, the "risk and restoration in probability" strategy is easier to implement and has better convergence performance since the best solution in history is always kept and restored frequently.

function ROTATION_FAST(funfcn,Best,SE,α,β))
  oldBest ← Best
  fBest ← feval(funfcn,oldBest)
  State ← op_roate_fast(Best,SE,α)
  [newBest,fnewBest] ← fitness(funfcn,State)
  if fnewBest < fBest **then**
    Best ← newBest
    fBest ← fnewBest
    State ← op_translate(oldBest,newBest,SE,β)
    [newBest,fnewBest] ← fitness(funfcn,State)
    if fnewBest < fBest **then**
      fBest ← fnewBest
      Best ← newBest
    end if
  else
    if *rand* < p_{risk} **then** ▷ risk in probability
      Best ← newBest
      fBest ← fnewBest
    end if
  end if
end function

4.4. Benchmark functions test

To evaluate the performance of the dynamic STA and to find a good combination of the restoration probability and the risk probability, some well-known benchmark functions are used for an empirical study, and they are listed below.

Spherical function

$$f_1 = \sum_{i=1}^{n} x_i^2, \quad x_i \in [0, 100],$$

Table 1
Parametric study of the dynamic STA (dimension = 100, iterations = 10,000).

Functions	p_{rest}	p_{risk}				
		0.1	0.3	0.5	0.7	0.9
Spherical	0.1	0 ± 0 ≈[a]	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	1.6231e−15 ± 5.1263e−15−
	0.3	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈
	0.5	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈
	0.7	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈
	0.9	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈
Rastrigin	0.1	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	3.3651e−12 ± 6.5316e−12−
	0.3	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	5.6843e−15 ± 2.5421e−14−
	0.5	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈
	0.7	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈
	0.9	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	1.1369e−14 ± 5.0842e−14−
Griewank	0.1	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	5.4956e−16 ± 1.3119e−15−
	0.3	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈
	0.5	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈
	0.7	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈
	0.9	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈	0 ± 0 ≈
Rosenbrock	0.1	34.9246 ± 48.8835−	80.9344 ± 29.7257−	83.5215 ± 19.7759−	87.8305 ± 0.3616−	88.0899 ± 0.2990−
	0.3	6.7223 ± 16.3449 ≈	32.5569 ± 45.5968−	94.9096 ± 43.6650−	111.6440 ± 56.4694−	106.1008 ± 49.9611−
	0.5	6.2421 ± 17.6002 ≈	2.1033 ± 2.6610+	28.3665 ± 54.6751−	95.1745 ± 73.8811−	91.6837 ± 77.8627−
	0.7	6.7115 ± 15.0144 ≈	16.8595 ± 34.9808−	16.2994 ± 36.4491−	24.7477 ± 35.8102−	20.8548 ± 51.9346−
	0.9	3.1765 ± 3.0737+	6.0835 ± 18.3434 ≈	6.2828 ± 18.1138 ≈	1.7381 ± 2.1587+	31.9860 ± 61.3152−
Ackley	0.1	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈	2.8025e−9 ± 4.4539e−9−
	0.3	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈
	0.5	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈
	0.7	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈
	0.9	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈	−8.8818e−16 ± 0 ≈
Methods		Spherical	Rastrigin	Griewank	Rosenbrock	Ackley
STA		0 ± 0 [b]	0 ± 0	0 ± 0	6.7952 ± 7.2544	−8.8818e−16 ± 0

[a] +, − and ≈ denote that the performance of corresponding algorithm is better than, worse than, and similar to that of the STA by the Wilcoxon rank sum test.
[b] ± denotes "mean ± standard deviation".

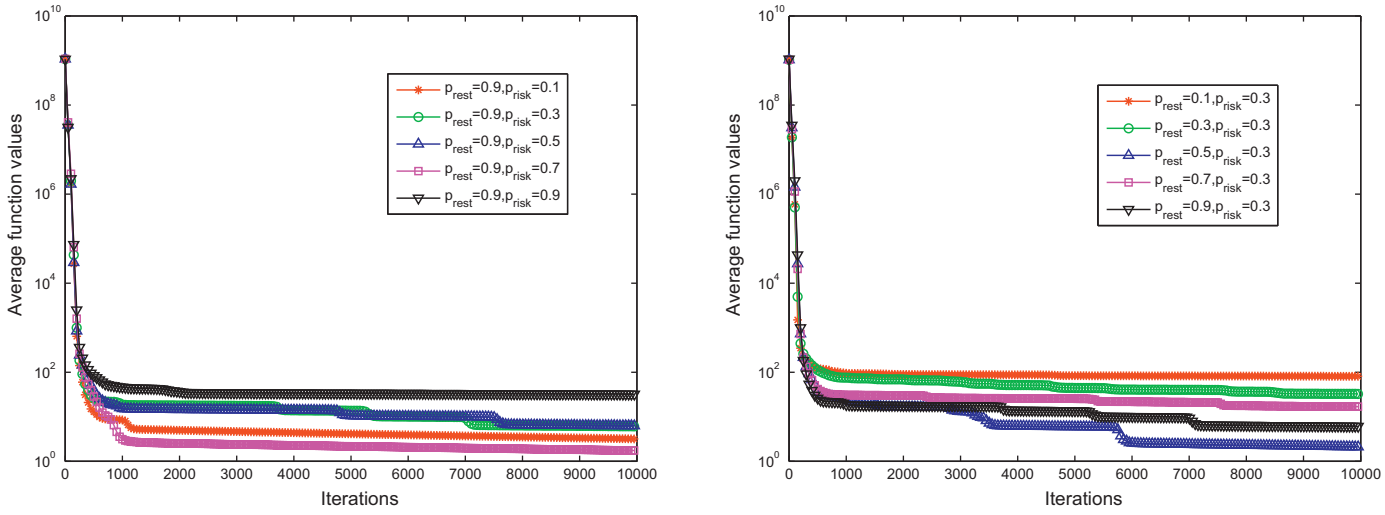

Fig. 7. Iterative curves of the dynamic STA with fixed "risk probability" $p_{rest} = 0.9$ or "restoration probability" $p_{risk} = 0.3$.

Rastrigin function

$$f_2 = \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10), \quad x_i \in [0, 5.12],$$

Griewank function

$$f_3 = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i}^{n}\cos\left|\frac{x_i}{\sqrt{i}}\right| + 1, \quad x_i \in [0, 600],$$

Rosenbrock function

$$f_4 = \sum_{i=1}^{n}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2), \quad x_i \in [0, 30],$$

Ackley function

$$f_5 = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e, \quad x_i \in [0, 32].$$

To investigate the effect of parameters in the dynamic adjustment strategy of the STA, we arranged empirical studies in order to find a satisfactory combination of p_{rest} and p_{risk}. First, with fixed dimension (problem size) and maximum number of iterations, some experimental tests are performed on the benchmark functions with different groups of (p_{rest}, p_{risk}). Table 1 shows the com-

Table 2
Performance of the dynamic STA for Rosenbrock function before refinement (dimension = 100).

Iterations	p_{rest}	p_{risk}				
		0.1	0.3	0.5	0.7	0.9
10,000	0.1	34.9246 ± 48.8835−	80.9344 ± 29.7257−	83.5215 ± 19.7759−	87.8305 ± 0.3616−	88.0899 ± 0.2990−
	0.3	6.7223 ± 16.3449 ≈	32.5569 ± 45.5968−	94.9096 ± 43.6650−	111.6440 ± 56.4694−	106.1008 ± 49.9611−
	0.5	6.2421 ± 17.6002 ≈	2.1033 ± 2.6610+	28.3665 ± 54.6751−	95.1745 ± 73.8811−	91.6837 ± 77.8627−
	0.7	6.7115 ± 15.0144 ≈	16.8595 ± 34.9808−	16.2994 ± 36.4491−	24.7477 ± 35.8102−	20.8548 ± 51.9346−
	0.9	3.1765 ± 3.0737+	6.0835 ± 18.3434 ≈	6.2828 ± 18.1138 ≈	1.7381 ± 2.1587+	31.9860 ± 61.3152−
20,000	0.1	36.6229 ± 41.1742−	70.7605 ± 30.1326−	80.3873 ± 18.9506−	84.8749 ± 0.2333−	85.4611 ± 0.2593−
	0.3	1.3527 ± 1.6541+	22.8921 ± 41.2240−	62.3492 ± 58.6085−	84.2958 ± 25.5081−	100.3479 ± 44.1913−
	0.5	6.7016 ± 18.1658−	1.3524 ± 1.6784+	21.9584 ± 47.6510−	48.4355 ± 58.3758−	69.9912 ± 63.0667−
	0.7	1.8801 ± 2.0266+	6.4236 ± 19.8274−	6.8750 ± 21.5073−	2.2362 ± 1.8067+	21.3934 ± 42.2174−
	0.9	2.1100 ± 1.5590+	1.9741 ± 1.9684+	2.2778 ± 1.8401+	11.8520 ± 34.9652−	18.9113 ± 38.2404−
30,000	0.1	8.1871 ± 22.3389−	59.4579 ± 34.4171−	81.9436 ± 0.8132−	82.8267 ± 0.2116−	83.5295 ± 0.2513−
	0.3	2.4303 ± 3.5954 ≈	30.1052 ± 44.2706−	59.8675 ± 39.3182−	83.9215 ± 39.0975−	84.6498 ± 14.2251−
	0.5	1.0879 ± 1.3156+	8.7277 ± 20.7838−	18.0923 ± 41.2037−	61.8106 ± 70.4271−	74.3550 ± 64.2713−
	0.7	0.7939 ± 0.8984+	1.3687 ± 1.1653+	9.9566 ± 24.4415−	10.3543 ± 27.5641−	33.5537 ± 49.5459−
	0.9	1.2680 ± 1.1676+	1.2800 ± 1.1730+	1.0241 ± 1.1646+	0.7466 ± 0.9355+	1.3722 ± 1.0550+
40,000	0.1	15.5817 ± 27.3889−	57.2248 ± 33.4997−	75.8210 ± 17.6466−	81.0726 ± 0.3210−	81.9298 ± 0.2200−
	0.3	0.6875 ± 0.8192+	9.3453 ± 23.8299−	46.0035 ± 46.7455−	69.6654 ± 39.9633−	84.8464 ± 21.9926−
	0.5	1.0538 ± 1.1154+	13.0401 ± 29.9575−	11.1267 ± 27.4796−	17.6139 ± 30.8468−	46.1817 ± 47.4052−
	0.7	0.9147 ± 0.6730+	1.3028 ± 0.9836 ≈	0.5366 ± 0.7615+	0.7609 ± 0.8081+	27.7787 ± 46.9281−
	0.9	1.0351 ± 0.8745+	0.7861 ± 0.7066+	0.5277 ± 0.8042+	1.2816 ± 0.7297 ≈	10.2427 ± 29.4908−
50,000	0.1	3.9619 ± 14.1378−	64.2790 ± 27.5534−	77.8868 ± 0.3830−	79.5461 ± 0.2291−	80.5820 ± 0.2589−
	0.3	0.7379 ± 0.7578+	15.7587 ± 30.7724−	26.7952 ± 36.4140−	42.4749 ± 36.2685−	79.6220 ± 37.3989−
	0.5	0.5307 ± 0.5997+	0.5488 ± 0.7058+	4.4704 ± 16.5813−	26.8494 ± 42.6035−	66.1161 ± 56.9610−
	0.7	0.6374 ± 0.5889+	0.5150 ± 0.5831+	5.5514 ± 14.3693−	0.6901 ± 0.9297+	10.1952 ± 25.8581−
	0.9	0.4749 ± 0.5146+	0.6032 ± 0.6193+	0.6189 ± 0.5609+	1.3870 ± 3.8526−	1.2649 ± 2.1263−
Methods		10,000	20,000	30,000	40,000	50,000
STA		6.7952 ± 7.2544	5.7962 ± 2.8367	2.8918 ± 1.9632	1.6408 ± 1.1822	1.0805 ± 0.7956

+, − and ≈ denote that the performance of corresponding algorithm is better than, worse than, and similar to that of the STA by the Wilcoxon rank sum test
± denotes "mean ± standard deviation".

Table 3
Performance of the dynamic STA for Rosenbrock function after refinement (dimension = 100).

Iterations	p_{rest}	p_{risk}				
		0.1	0.3	0.5	0.7	0.9
10,000	0.1	21.1587 ± 36.1116−	53.1184 ± 29.5350−	63.3643 ± 20.1991−	70.5142 ± 1.3252−	70.2448 ± 1.0381−
	0.3	0.0204 ± 0.0912 ≈	7.5377 ± 22.0972−	42.4304 ± 38.3322−	50.5568 ± 42.2048−	60.5727 ± 31.6022−
	0.5	0.2802 ± 1.2511−	4.0550e−4 ± 0.0018+	5.7047 ± 17.5013−	51.1806 ± 54.7261−	47.0276 ± 46.6980−
	0.7	5.7870e−4 ± 0.0026+	2.6333e−6 ± 8.7040e−6+	9.7799e−4 ± 0.0044+	10.3570 ± 25.3780−	10.3748 ± 25.3956−
	0.9	0.0013 ± 0.0057 ≈	5.7115e−5 ± 2.5542e−4+	0.0920 ± 0.4115 ≈	9.3007e−10 ± 1.4912e−9+	4.2359 ± 16.8367−
20,000	0.1	17.6076 ± 31.7114−	57.2231 ± 24.6991−	64.7344 ± 15.3070−	65.4911 ± 13.0773−	68.9557 ± 1.0903−
	0.3	1.2964e−9 ± 1.7722e−9+	4.5672e−9 ± 1.5513e−8−	27.0481 ± 33.2740−	53.1332 ± 26.1597−	65.3754 ± 24.7380−
	0.5	3.3265 ± 14.8456−	9.1372e−9 ± 3.6650e−8+	5.4289 ± 18.2178−	25.2178 ± 41.7371−	28.3450 ± 36.0765−
	0.7	1.7181e−9 ± 2.9007e−9+	0.2978 ± 1.3318−	3.9112 ± 17.4915−	2.3355e−9 ± 2.3355e−9+	4.3373 ± 19.3952−
	0.9	2.5848e−9 ± 4.0031e−9+	6.4586e−9 ± 2.4736e−8+	9.7906e−10 ± 1.4118e−9+	1.8054 ± 7.5612−	7.2380e−8 ± 3.2018e−7+
30,000	0.1	5.8828 ± 18.1071−	45.2268 ± 30.3826−	66.6607 ± 0.9523−	66.9913 ± 0.9594−	67.1216 ± 1.0454−
	0.3	0.1111 ± 0.4969−	8.9327 ± 21.3730−	28.6617 ± 36.1546−	53.0224 ± 27.8523−	61.5713 ± 18.5342−
	0.5	9.1503e−10 ± 1.4908e−9 ≈	2.8200 ± 12.6115−	10.7349 ± 25.6296−	21.6996 ± 32.4225−	32.2821 ± 41.0405−
	0.7	4.9650e−10 ± 5.4828e−10 ≈	1.0867e−9 ± 1.4004e−9 ≈	0.2918 ± 1.3052−	3.8141 ± 13.5439−	7.9870 ± 22.5467−
	0.9	1.6998e−9 ± 2.7782e−9 ≈	1.4478e−9 ± 2.1626e−9 ≈	8.5037e−10 ± 1.5985e−9 ≈	2.4864e−7 ± 1.1101e−6−	2.2585e−9 ± 6.0799e−9 ≈
40,000	0.1	10.7407 ± 21.8229−	40.3449 ± 30.3853−	61.1384 ± 14.4172−	65.1246 ± 1.0444−	65.3149 ± 0.9135−
	0.3	4.7625e−10 ± 5.8946e−10 ≈	6.6588 ± 18.9926−	17.8645 ± 27.0329−	40.9452 ± 29.2316−	56.2615 ± 19.3012−
	0.5	1.2431e−9 ± 2.7342e−9 ≈	0.2716 ± 0.8660−	4.8668 ± 19.8535−	6.0495 ± 17.4021−	27.8799 ± 33.3700−
	0.7	9.2246e−10 ± 1.4431e−9 ≈	1.4476e−9 ± 2.2476e−9 ≈	7.7317e−10 ± 1.6800e−9 ≈	1.0940e−9 ± 1.8583e−9 ≈	6.9124 ± 17.7629−
	0.9	7.0819e−10 ± 7.4544e−10 ≈	8.6857e−10 ± 1.2423e−9 ≈	8.6832e−10 ± 1.3687e−9 ≈	1.4373e−9 ± 1.8758e−9 ≈	1.3033 ± 5.8287−
50,000	0.1	2.4878 ± 11.1259−	42.2141 ± 28.4066−	62.6582 ± 0.9551−	64.0278 ± 0.7651−	64.6633 ± 1.1308−
	0.3	7.3724e−10 ± 9.7763e−10 ≈	5.6969 ± 17.7139−	14.1997 ± 24.6097−	19.8874 ± 26.7939−	54.9271 ± 20.6629−
	0.5	4.7795e−10 ± 5.8795e−10 ≈	1.7176e−9 ± 3.1498e−9 ≈	3.2398e−9 ± 1.2368e−8 ≈	8.5287 ± 20.8636−	34.2486 ± 38.3122−
	0.7	2.0098e−9 ± 4.6715e−9 ≈	6.9608e−10 ± 6.2422e−10 ≈	3.0832 ± 10.7902−	1.5723e−9 ± 3.4520e−9 ≈	4.5651 ± 18.0369−
	0.9	9.7722e−10 ± 1.5721e−9 ≈	8.5754e−10 ± 8.6167e−10 ≈	9.6368e−10 ± 1.4701e−9 ≈	0.1362 ± 0.6091−	1.5806e−9 ± 2.4191e−9 ≈
Methods		10,000	20,000	30,000	40,000	50,000
STA		0.0806 ± 0.2253	3.1866e−4 ± 0.0011	3.0420e−9 ± 4.8701e−9	2.7837e−9 ± 4.8575e−9	9.7776e−10 ± 1.2594e−9

+, − and ≈ denote that the performance of corresponding algorithm is better than, worse than, and similar to that of the STA by the Wilcoxon rank sum test.
± denotes "mean ± standard deviation".

parison results between the dynamic STA and the basic STA. From Table 1, we observe that for most groups of (p_{rest}, p_{risk}), excluding $(p_{rest}, p_{risk}) = (0.1, 0.9)$ and $(p_{rest}, p_{risk}) = (0.3, 0.9)$, for all the benchmark functions except the Rosenbrock function, the statistical performance of the dynamic STA and the basic STA is almost the same when using the Wilcoxon rank sum test. However, for the Rosenbrock problem, the results of the dynamic STA with parameter groups $(p_{rest}, p_{risk}) = (0.3, 0.1)$, $(p_{rest}, p_{risk}) = (0.5, 0.1)$, $(p_{rest}, p_{risk}) = (0.7, 0.1)$, $(p_{rest}, p_{risk}) = (0.9, 0.1)$, $(p_{rest}, p_{risk}) = (0.5, 0.3)$, $(p_{rest}, p_{risk}) = (0.9, 0.3)$, $(p_{rest}, p_{risk}) = (0.9, 0.5)$ and $(p_{rest}, p_{risk}) = (0.9, 0.7)$ are better or similar to those obtained by the basic STA. For a fixed restoration probability $p_{rest} = 0.9$

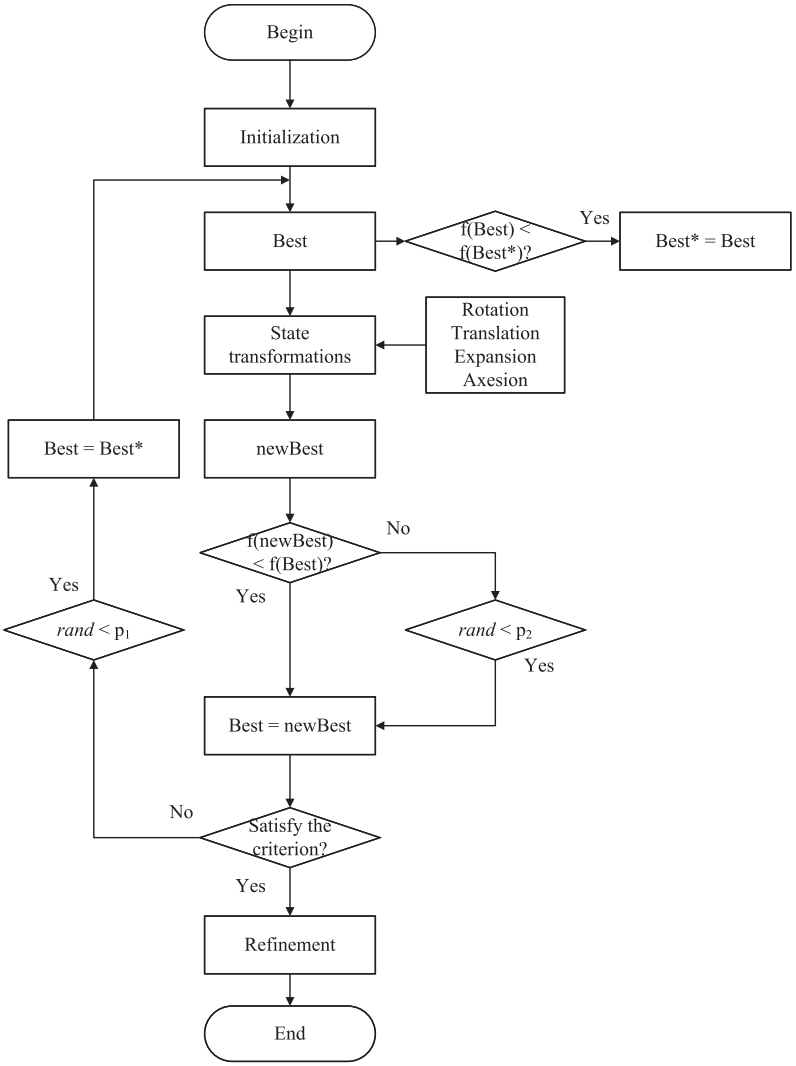

Fig. 8. The flowchart of the dynamic STA with refinement.

or a fixed risk probability $p_{\text{risk}} = 0.3$, the iterative curves of the dynamic STA with the setting parameters are given in Fig. 7. It can be seen that either p_{risk} or p_{rest} can influence the dynamic STA's performance.

To further investigate the effect of $(p_{\text{rest}}, p_{\text{risk}})$ on the Rosenbrock function, which belongs to the sum-of-squares problem, including the sensor network localization problem, we increase the maximum number of iterations from 100 to 500 times the dimension and refine the solutions obtained by incorporating a gradient-based method. As shown in Tables 2 and 3, the majority of the aforementioned parameter groups still perform well for the dynamic STA. The results after refinement are much closer to the true global solutions of the Rosenbrock problem, and the larger the maximum number of iterations, the closer to the true global solutions. After taking stability and reliability into consideration, the parameter group $(p_{\text{rest}}, p_{\text{risk}}) = (0.9, 0.3)$ is selected as the empirical best choice.

Remark 4. Through the benchmark function tests, the restoration probability p_{rest} and the risk probability p_{risk} in the dynamic STA are determined empirically. The parameter group $(p_{\text{rest}}, p_{\text{risk}}) = (0.9, 0.3)$ is also considered reasonable because it tells us that a risk should be taken at a low probability and that the historical best should be restored at a high probability. In addition, with a gradient-based technique incorporated for refinement, the flowchart of the dynamic STA with refinement is illustrated in Fig. 8.

Table 4
Parameters setting for STA and DSTA.

Methods	$\alpha(\beta, \gamma, \delta)$	fc	SE	(p_1, p_2)	Maxiter
STA	$1 \rightarrow 1e{-}8$	2	30	–	1000
DSTA	$1 \rightarrow 1e{-}8$	2	30	(0.9, 0.3)	1000

5. Application for the sensor network localization

In this section, the proposed dynamic STA with refinement is applied to solve the SNL problem. Firstly, an illustrative example is given to show the superiority of dynamic STA. Then, two large SNL problems are given to verify the effectiveness of the proposed approach. The network topology of the illustrative example is given in Fig. 9, where there are 4 anchors with known positions. We

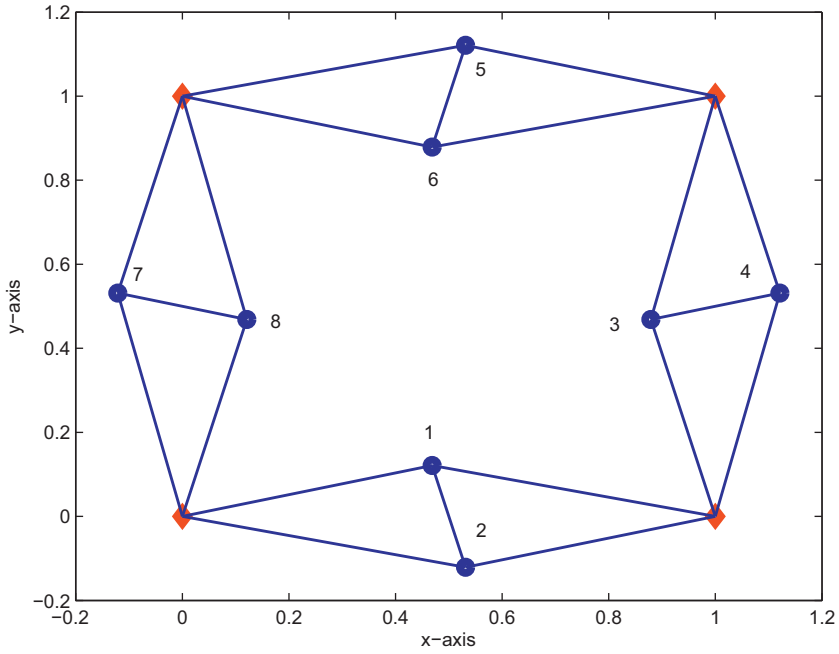

Fig. 9. Network topology of the illustrative example.

Table 5
Numerical results for 8 sensors and 4 anchors.

True solutions	Solutions by STA	Solutions by DSTA	Solutions by SFSDP
$\boldsymbol{x}_1^* = (0.4688, 0.1210)$	$\bar{\boldsymbol{x}}_1 = (0.4688, 0.1210)$	$\bar{\boldsymbol{x}}_1 = (0.4688, 0.1210)$	$\bar{\boldsymbol{x}}_1 = (0.4687, 0.0005)$
$\boldsymbol{x}_2^* = (0.5313, -0.1210)$	$\bar{\boldsymbol{x}}_2 = (0.5313, -0.1211)$	$\bar{\boldsymbol{x}}_2 = (0.5312, -0.1211)$	$\bar{\boldsymbol{x}}_2 = (0.5312, 0.0005)$
$\boldsymbol{x}_3^* = (0.8790, 0.4688)$	$\bar{\boldsymbol{x}}_3 = (0.8790, 0.4687)$	$\boldsymbol{x}_3^* = (0.8790, 0.4688)$	$\boldsymbol{x}_3^* = (0.9907, 0.4687)$
$\boldsymbol{x}_4^* = (1.1210, 0.5313)$	$\bar{\boldsymbol{x}}_4 = (1.1211, 0.5312)$	$\boldsymbol{x}_4^* = (1.1210, 0.5313)$	$\boldsymbol{x}_4^* = (0.9908, 0.5312)$
$\boldsymbol{x}_5^* = (0.5313, 1.1210)$	$\bar{\boldsymbol{x}}_5 = (0.5312, 1.1211)$	$\bar{\boldsymbol{x}}_5 = (0.5312, 1.1210)$	$\bar{\boldsymbol{x}}_5 = (0.5312, 0.9908)$
$\boldsymbol{x}_6^* = (0.4688, 0.8790)$	$\bar{\boldsymbol{x}}_6 = (0.4688, 0.8790)$	$\bar{\boldsymbol{x}}_6 = (0.4688, 0.8790)$	$\bar{\boldsymbol{x}}_6 = (0.4687, 0.9907)$
$\boldsymbol{x}_7^* = (-0.1210, 0.5313)$	$\bar{\boldsymbol{x}}_7 = (-0.1210, 0.5313)$	$\bar{\boldsymbol{x}}_7 = (-0.1210, 0.5312)$	$\bar{\boldsymbol{x}}_7 = (0.0005, 0.5312)$
$\boldsymbol{x}_8^* = (0.1210, 0.4688)$	$\bar{\boldsymbol{x}}_8 = (0.1210, 0.4687)$	$\boldsymbol{x}_8^* = (0.1210, 0.4688)$	$\boldsymbol{x}_8^* = (0.0005, 0.4687)$
Methods	min	mean	std. dev.
STA	8.1301e−10	2.5723e−9	1.4557e−9
DSTA	4.2838e−18	2.3756e−17	2.3502e−17
SFSDP	0.0171	0.0171	0

Table 6
Numerical results for larger instances.

Instances	Methods	min	mean	std. dev.
50 sensors and 4 anchors	STA	2.3129e−5	0.0014	0.0041
	DSTA	2.3129e−5	2.3129e−5	1.2280e−11
250 sensors and 4 anchors	STA	1.8733e−11	1.5232e−8	5.1607e−8
	DSTA	1.6482e−11	5.3354e−11	3.2959e−11

need to decide the positions of 8 sensors such that the following distances between two sensors or between a sensor and an anchor are satisfied:

$$\boldsymbol{a}_1 = (0,0), \boldsymbol{a}_2 = (0,1), \boldsymbol{a}_3 = (1,0), \boldsymbol{a}_4 = (1,1),$$
$$d_{12} = 1/4, d_{34} = 1/4, d_{56} = 1/4, d_{78} = 1/4,$$
$$e_{11} = \frac{\sqrt{15}}{8}, e_{13} = \frac{\sqrt{19}}{8}, e_{21} = \frac{\sqrt{19}}{8}, e_{23} = \frac{\sqrt{15}}{8},$$
$$e_{33} = \frac{\sqrt{15}}{8}, e_{34} = \frac{\sqrt{19}}{8}, e_{43} = \frac{\sqrt{19}}{8}, e_{44} = \frac{\sqrt{15}}{8},$$
$$e_{52} = \frac{\sqrt{19}}{8}, e_{54} = \frac{\sqrt{15}}{8}, e_{62} = \frac{\sqrt{15}}{8}, e_{64} = \frac{\sqrt{19}}{8},$$
$$e_{71} = \frac{\sqrt{19}}{8}, e_{72} = \frac{\sqrt{15}}{8}, e_{81} = \frac{\sqrt{15}}{8}, e_{82} = \frac{\sqrt{19}}{8}.$$

For the illustrative example, the parameters used for the STA and dynamic STA (DSTA) are given in Table 4. The search enforcement $SE = 30$, the transformation factors are all decreasing exponentially from 1 to $1e-8$ with the base $fc = 2$, and the maximum number of iterations (Maxiter) is 1000. The programs that implement the STA and DSTA are run for 20 trials independently in MATLAB R2010b on a PC with Intel(R) Core(TM) i3-2310M CPU @2.10GHz under Windows 7. The embedded built-in function *fminunc* is used as a gradient-based optimization technique for refinement. To evaluate the performance of the proposed approach, we introduce a sparse version of full semi-definite programming (SFSDP), which is a Matlab package for solving sensor network localization problems, for performance comparison. Numerical results are given in Table 5. We observe that both STA and DSTA can find the true global solution. In addition, DSTA is better than STA due to its enhanced exploitation ability, which can be verified by

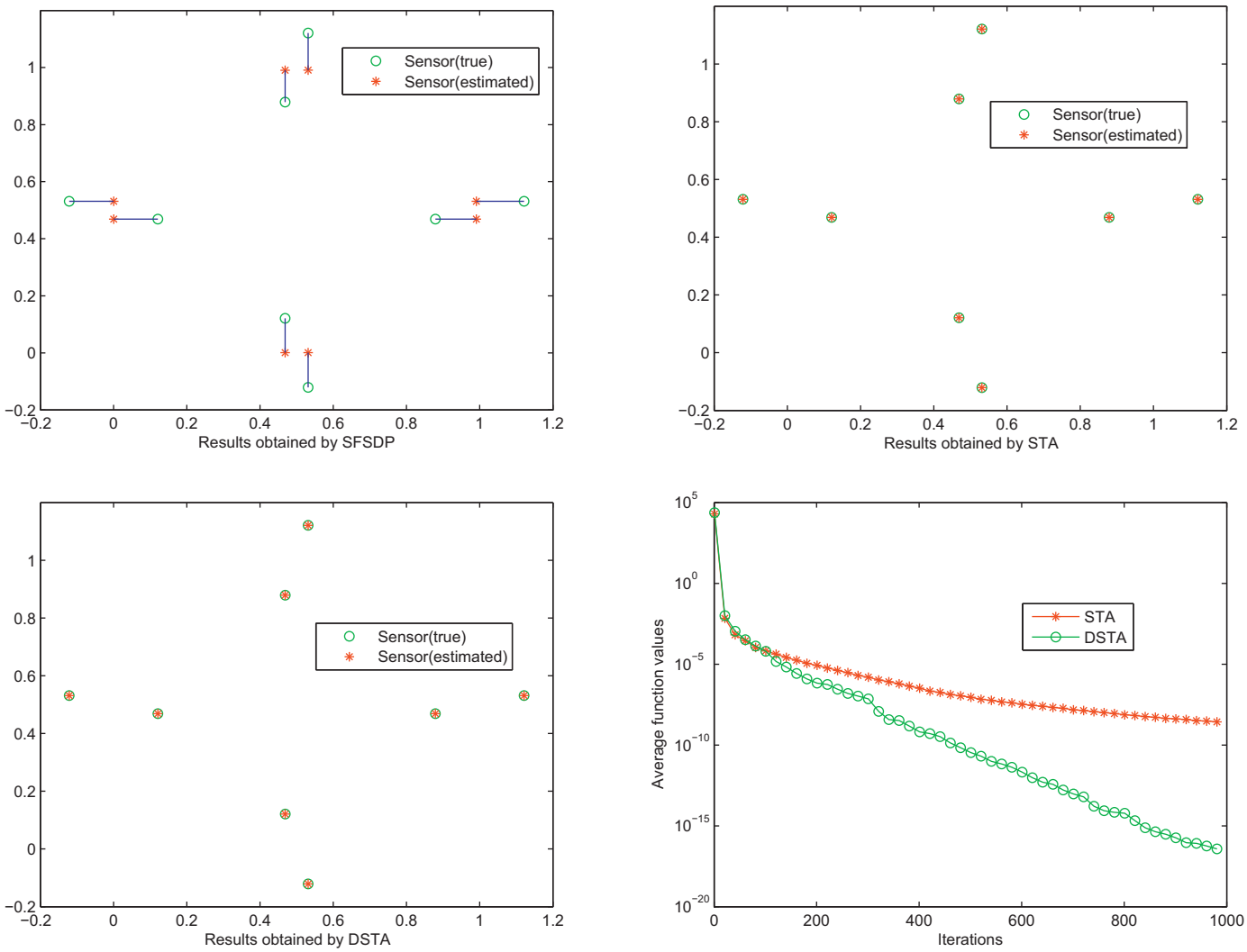

Fig. 10. Performance comparison of three different algorithms for the illustrative example.

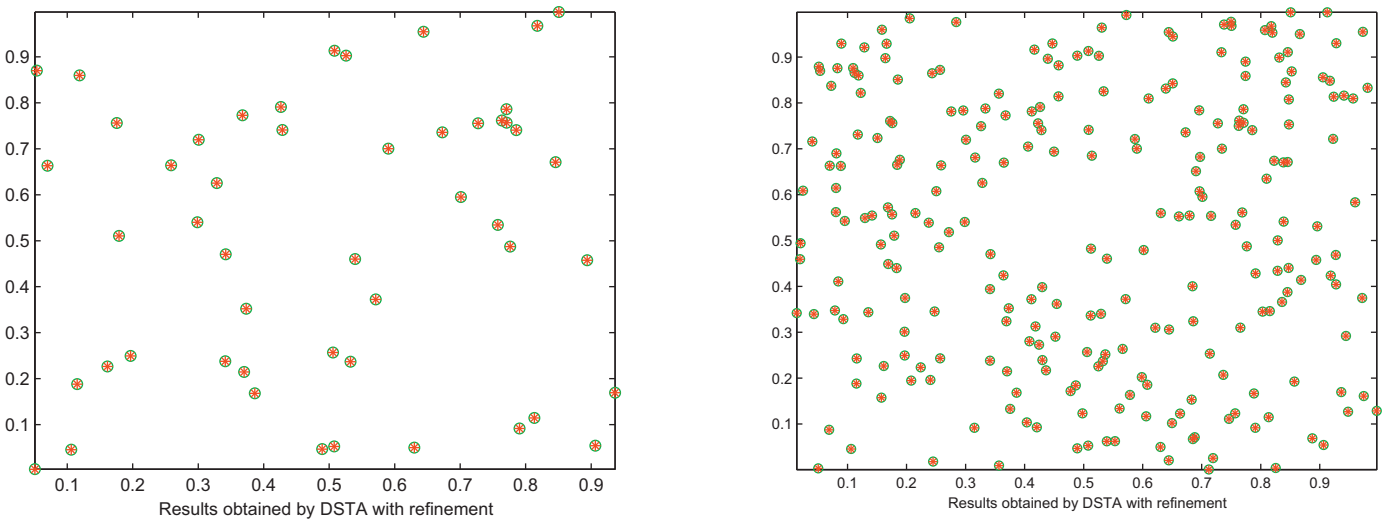

Fig. 11. Best results obtained by dynamic STA with refinement for the SNL problem with 50 and 250 sensors, respectively.

the statistics (min, mean, etc) and the iterative curves of the average function values in Fig. 10. Note that the SFSDP based on gradients cannot find the true global solution.

Next, two larger sensor network localization problems are investigated. The first problem contains 50 sensors and 4 anchors. The second contains 250 sensors and 4 anchors. A radio range of 0.3 is used. The locations of the sensors and the anchors are randomly created by the benchmark generator embedded in the Matlab package. In addition, some noise is added to both problems with noisy factor 0.001.

The experimental results are given in Table 6 and Fig. 11. Both STA and dynamic STA can find the global solutions for the two problems, with the performance of dynamic STA being much better due to its more stable results as indicated by the std.dev.

6. Conclusion

By using the least squares method, the SNL problem can be reformulated as a non-convex optimization problem. In this paper, we have proposed a dynamic STA with refinement for the SNL problem. A dynamic adjustment strategy called risk and restoration in probability has been incorporated into the basic STA for transcending local optimality and improving its global search ability. Monte Carlo experiments have been designed to obtain a satisfactory combination of the risk probability and restoration probability. With the gained parameters setting, the DSTA have been successfully applied to some instances of the SNL problem with a problem size as large as 500. Numerical results have shown that the proposed DSTA has better performance compared with the basic STA in terms of global search ability and solution quality, which has verified the effectiveness and efficiency of the proposed approach. In our future work, to better locate unknown sensors for the SNL problem in real-world applications, it is advantageous to use other techniques to determine more anchor sensors in advance. Furthermore, other strategies to accelerate the search process of STA is preferred.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Grant No. 61503416, 61533020, 61590921, 61673399), the Australian Research Council (DP140102180, LP140100471) and the 111 Project (B12018, B17048).

References

[1] G. Serpen, L. Liu, Parallel and distributed neurocomputing with wireless sensor networks, Neurocomputing 173 (2016) 1169–1182.
[2] H.M. La, W. Sheng, Distributed sensor fusion for scalar field mapping using mobile sensor networks, IEEE Trans. Cybern. 43 (2) (2013) 766–778.
[3] W. Yang, H. Shi, Sensor selection schemes for consensus based distributed estimation over energy constrained wireless sensor networks, Neurocomputing 87 (2012) 132–137.
[4] G. Mao, B. Fidan, B. Anderson, Wireless sensor network localization techniques, Comput. Netw. 51 (10) (2007) 2529–2553.
[5] X. Wang, S. Wang, D. Bi, Distributed visual-target-surveillance system in wireless sensor networks, IEEE Trans. Syst. Man Cybern. Part B Cybern. 39 (5) (2009) 1134–1146.
[6] H. Mohamadi, S. Salleh, M.N. Razali, S. Marouf, A new learning automata-based approach for maximizing network lifetime in wireless sensor networks with adjustable sensing ranges, Neurocomputing 153 (2015) 11–19.
[7] Y. Yoon, Y.-H. Kim, An efficient genetic algorithm for maximum coverage deployment in wireless sensor networks, IEEE Trans. Cybern. 43 (5) (2013) 1473–1483.
[8] Y. Luo, Z. Wang, G. Wei, F.E. Alsaadi, T. Hayat, State estimation for a class of artificial neural networks with stochastically corrupted measurements under round-robin protocol, Neural Netw. 77 (2016) 70–79.
[9] Y. Luo, Z. Wang, G. Wei, F.E. Alsaadi, H_∞ fuzzy fault detection for uncertain 2-d systems under round-robin scheduling protocol, IEEE Trans. Syst. Man Cybern. Syst. 47 (8) (2017) 2172–2184.
[10] T.-C. Liang, T.-C. Wang, Y. Ye, A gradient search method to round the semidefinite programming relaxation solution for ad hoc wireless sensor network localization, Formal Report, Stanford University 5, 2004.
[11] J. Aspnes, D. Goldenberg, Y.R. Yang, On the computational complexity of sensor network localization, in: Proceedings of the Algorithmic Aspects of Wireless Sensor Networks, Springer, 2004, pp. 32–44.
[12] P. Biswas, Y. Ye, Semidefinite programming for ad hoc wireless sensor network localization, in: Proceedings of the Third International Symposium on Information Processing in Sensor Networks, ACM, 2004, pp. 46–54.
[13] A.M.-C. So, Y. Ye, Theory of semidefinite programming for sensor network localization, Math. Programm. 109 (2–3) (2007) 367–384.
[14] Z. Wang, S. Zheng, Y. Ye, S. Boyd, Further relaxations of the semidefinite programming approach to sensor network localization, SIAM J. Optim. 19 (2) (2008) 655–673.
[15] S. Srirangarajan, A.H. Tewfik, Z.-Q. Luo, Distributed sensor network localization using SOCP relaxation, IEEE Trans. Wireless Commun. 7 (12) (2008) 4886–4895.
[16] P. Tseng, Second-order cone programming relaxation of sensor network localization, SIAM J. Optim. 18 (1) (2007) 156–185.
[17] M.W. Carter, H.H. Jin, M.A. Saunders, Y. Ye, Spaseloc: an adaptive subproblem algorithm for scalable wireless sensor network localization, SIAM J. Optim. 17 (4) (2006) 1102–1128.
[18] J. Nie, Sum of squares method for sensor network localization, Comput. Optim. Appl. 43 (2) (2009) 151–179.
[19] N. Ruan, D.Y. Gao, Global optimal solutions to general sensor network localization problem, Perform. Eval. 75 (2014) 1–16.
[20] C. Wu, C. Li, D.Y. Gao, Canonical primal-dual method for solving non-convex minimization problems, arXiv:1212.6492 (2012).
[21] X. Zhou, D.Y. Gao, C. Yang, Canonical primal-dual algorithm for solving fourth-order polynomial minimization problems, Appl. Math. Comput. 227 (2014) 246–255.
[22] A. Gopakumar, L. Jacob, Performance of some metaheuristic algorithms for localization in wireless sensor networks, Int. J. Netw. Manag. 19 (5) (2009) 355–373.
[23] R.V. Kulkarni, G.K. Venayagamoorthy, Particle swarm optimization in wireless-sensor networks: a brief survey, IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. 41 (2) (2011) 262–267.
[24] K.-S. Low, H. Nguyen, H. Guo, A particle swarm optimization approach for the localization of a wireless sensor network, in: Proceedings of the IEEE International Symposium on Industrial Electronics, IEEE, 2008, pp. 1820–1825.
[25] X. Zhou, C. Yang, W. Gui, State transition algorithm, J. Indust. Manag. Optim. 8 (4) (2012) 1039–1056.
[26] X. Zhou, D.Y. Gao, C. Yang, A comparative study of state transition algorithm with harmony search and artificial bee colony, Adv Intell. Syst. Comput. 213 (2013) 651–659.
[27] X. Zhou, C. Yang, W. Gui, Nonlinear system identification and control using state transition algorithm, Appl. Math. Comput. 226 (2014) 169–179.
[28] X. Zhou, C. Yang, W. Gui, A comparative study of sta on large scale global optimization, in: Proceedings of the Twelfth World Congress on Intelligent Control and Automation (WCICA), IEEE, 2016, pp. 2115–2119.
[29] X. Zhou, D.Y. Gao, C. Yang, W. Gui, Discrete state transition algorithm for unconstrained integer optimization problems, Neurocomputing 173 (2016) 864–874.
[30] F. Zhang, C. Yang, X. Zhou, W. Gui, Fractional-order PID controller tuning using continuous state transition algorithm, Neural Comput. Appl. (2016) 1–10.
[31] X. Zhou, D.Y. Gao, A.R. Simpson, Optimal design of water distribution networks by a discrete state transition algorithm, Eng. Optim. 48 (4) (2016) 603–628.
[32] J. Han, C. Yang, X. Zhou, W. Gui, A new multi-threshold image segmentation approach using state transition algorithm, Appl. Math. Modell. 44 (2017) 588–601.
[33] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simmulated annealing, Science 220 (4598) (1983) 671–680.
[34] D. Mitra, F. Romeo, A. Sangiovanni-Vincentelli, Convergence and finite-time behavior of simulated annealing, in: Proceedings of the Twenty-fourth IEEE Conference on Decision and Control, 24, IEEE, 1985, pp. 761–767.

Xiaojun Zhou received his Bachelor's degree in Automation in 2009 from Central South University, Changsha, China and received the Ph.D. degree in Applied Mathematics in 2014 from Federation University Australia. He is currently an Associate Professor at Central South University, Changsha, China. His main interests include modeling, optimization and control of complex industrial process, optimization theory and algorithms, state transition algorithm, duality theory and their applications.

Peng Shi received the B.Sc. degree in mathematics from the Harbin Institute of Technology, Harbin, China, the M.E. degree in systems engineering from Harbin Engineering University, Harbin, the Ph.D. degree in electrical engineering from the University of Newcastle, Callaghan, NSW, Australia, the Ph.D. degree in mathematics from the University of South Australia, Adelaide, SA, Australia, the Doctor of Science degree from the University of Glamorgan, Pontypridd, U.K., in 2006, and the Doctor of Engineering degree from the University of Adelaide in 2015. He was a Professor with the University of Glamorgan, a Senior Scientist with the Defence Science and Technology Organisation, Adelaide, Australia, a Lecturer and Post-Doctorate with the University of South Australia. He is currently a Professor with the University of Adelaide, and Victoria University, Melbourne, VIC, Australia. His current research interests include system and control theory, computational intelligence, and operational research. He has actively served as an Editorial Board Member of a number of journals, including Automatica, the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, the IEEE TRANSACTIONS ON FUZZY SYSTEMS, the IEEE TRANSACTIONS ON CYBERNETICS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I: REGULAR PAPERS, and the IEEE ACCESS. He was the Chair of Control Aerospace and Electronic Systems Chapter, IEEE South Australia Section. He is currently a member of the College of Expert, Australian Research Council. He is a fellow of the Institution of Engineering and Technology and the Institute of Mathematics and its Applications.

Cheng-Chew Lim received the B.Sc. (Hons.) and Ph.D. degrees in electronic and electrical engineering from Loughborough University, Leicestershire, U.K. He is a Professor of Electrical and Electronic Engineering and the former Head of the School of Electrical and Electronic Engineering, University of Adelaide, Adelaide, SA, Australia. His current research interests include control systems and optimization techniques and applications. He is serving as an Editorial Board Member for the Journal of Industrial and Management Optimization. He has served as a Guest Editor of a number of journals, including Discrete and Continuous Dynamical System—Series B and the Chair of the IEEE Chapter on Control and Aerospace Electronic Systems, IEEE South Australia Section.

Chunhua Yang received her M.Eng. in Automatic Control Engineering and her Ph.D. in Control Science and Engineering from Central South University, China in 1988 and 2002, respectively, and was with the Electrical Engineering Department, Katholieke Universiteit Leuven, Belgium from 1999 to 2001. She is currently a full professor in the School of Information Science & Engineering, Central South University. Her research interests include modeling and optimal control of complex industrial process, intelligent control system, and fault-tolerant computing of real-time systems.

Weihua Gui received the degree of the B.Eng. (Automatic Control Engineering) and the M.Eng. (Control Science and Engineering) from Central South University, Changsha, China in 1976 and 1981, respectively. From 1986 to 1988, he was a visiting scholar at Universitat-GH-Duisburg Germany. He is a member of the Chinese Academy of Engineering and has been a full professor in the School of Information Science & Engineering, Central South University, Changsha, China, since 1991. His main research interests are in modeling and optimal control of complex industrial process, distributed robust control, and fault diagnoses.