SWARM INTELLIGENCE OVER RANDOM NETWORKS

FEI CHEN, ZENGQIANG CHEN, LINYING XIANG, ZHONGXIN LIU AND ZHUZHI YUAN

Abstract. The random network model is incorporated in an optimization algorithm (SIRN), where each code representing the solution to an optimization problem is denoted by a node and the relations among codes are described by a time-varying network that is derived from a directed version of Erdős and Rényi (ER) model. It is shown that the inherent random network improves the performance of SIRN as compared with some traditional optimization algorithms, e.g., genetic algorithm (GA) where no network is employed.

Key Words. random network, ER model, complex network, multi-agent system and genetic algorithm.

1. Introduction

Complexity is an inherently interdisciplinary concept that has penetrated a range of intellectual fields from physics to cybernetics, but with no underlying, unified theory[1]. In fact, every complex system in the world is made of many highly interconnected parts on many scales, the interactions among which can be modeled by networks[2,3]. People are often interested in revealing the principles, e.g., the network patterns that govern the ways in which a new feature emerges[4].

Complex weblike structures describe a wide variety of systems of high technological and intellectual importance[2,5], e.g., the Internet and the World Wide Web. Traditionally, the topology of a large networked system is described by a completely random graph, the so-called Erdős-Rényi (ER) random model[5]. The theory of random graphs was founded by Paul Erdős and Alfréd Rényi, after Erdős discovered that probabilistic methods were often useful in tackling problems in graph theory. Then Watts, Strogatz, Newman, and Monasson introduced the small-world network models, which have short average path lengths along with large clusters[6]. Another significant discovery is the scale-free features in a number of actual networking systems, whose degree distributions follow power laws as first pointed out by Barabási and Albert[7].

In addition, the dynamic of networks has also attracted tremendous attention from various fields, for instance, the control and synchronization of complex networks, where particular interest is devoted to understanding how the dynamic of networks depends on various structural parameters of the networks, such as average distance, clustering coefficient, and degree distribution among others[8]. In the past few years, great achievement has been made in developing the theory of complex networks, which has been applied extensively to various disciplines. Perhaps this
paper is such an example since it aims at giving an optimization algorithm that
takes the ideas from the theory of complex networks.

Next, the basic knowledge of optimization problems is provided. Optimization
problems are made up of three basic ingredients:

- An objective function which needs to be minimized or maximized. For
  instance, in a manufacturing process, one may wish to minimize the overall
  completion time.
- A set of variables which determine the value of the objective function. In
  the manufacturing problem, the variables may include the times spent on
  activities.
- A set of constraints that make the variables take on certain values but
  exclude others. For the manufacturing problem, a negative amount of time
  on any activity makes no sense, so one may constrain all the “time” variables
  to be non-negative.

The optimization problem is then: Find values of the variables that minimize or
maximize the objective function while satisfying the constraints[9].

In this paper, the random network model is introduced in an algorithm for
optimization (SIRN). Each code (See Sec. 5.1) is encoded as a possible solution
and the relations among codes are described by a network that is derived from a
directed version of Erdős and Rényi (ER) model. Codes evolve using local rules
based on the information from their “neighbors”. It is analyzed that by following a
simple local law the swarm is able to find globally optimal solutions almost surely.
In addition, by numerous numerical examples, it is shown that the inherent random
network helps SIRN keep its diversity and as a result leads to better performance
as compared with some traditional optimization algorithms, e.g., genetic algorithm
(GA) where no network is employed.

2. Random network models

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{Illustration of the graph evolution process for Erdős-Rényi model.}
\end{figure}

In SIRN, each code is represented by a vector. The interactions between two
codes are represented by an arc, i.e., if code $i$ receives information from code $j$ there
is an arc from code $i$ to code $j$. In this way, the interactions among codes can be
modeled by a network. In current paper, the networks are derived from a directed
version of ER model and codes evolve by following local rules which depend solely
on the states of their neighbors. In the following, ER model is first introduced.

In their classic first article on random graphs [5], Erdős and Rényi defined a
random graph as $n$ labeled nodes connected by edges which are chosen randomly
from the $\frac{n(n-1)}{2}$ possible edges with a fixed probability $p$. In total there are $\left(\frac{n(n-1)}{2}\right)$
graphs with \( n \) nodes and \( k \) edges, forming a probability space in which every realization is equiprobable. Fig. 1 illustrates the evolution of Erdős-Rényi graphs, where the number of nodes \( n = 20 \) and the probability \( p \) is 0, 0.05, and 0.10 respectively. The interested reader is referred to [5]. ER model can easily be extended to the directed case by requiring that each arc appears with a fixed probability \( p \). In SIRN, at each time, the network describing the interactions among codes is derived from the directed version of ER model. It is noteworthy that the network generated by the directed version of ER model may not be connected.

3. The SIRN Algorithm

For convenience, some basic terms of SIRN are first introduced.

3.1. Glossary. Code: A code \( c_i \), which is a vector of binary entries, represents a possible solution to an optimization problem. The length of a code is defined to be the number of entries in the code. Fig. 2 demonstrates the possible form of a code.

![Figure 2. A possible form of a code with length \( l = 6 \).](image)

Population: A set of codes. The number of codes, denoted by \( n \), in a population is called the size of this population.

Fitness function: A function evaluates the fitness of a code and gives a score (fitness) based on how well it performs at a given task. The fitness of code \( c_i \) is denoted by \( f_i \).

Diversity: A key feature of an efficient and reliable optimization algorithm is the ability to maintain swarm diversity within a population of solutions. In this note, the diversity \( d \) is defined as

\[
d = \left( \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \| c_i - c_j \|^2 \right)^{\frac{1}{2}},
\]

where \( n \) is the size of population and \( \| \cdot \| \) indicates the Euclidian norm. It is straightforward to see \( 0 \leq d \leq 1 \). The diversity in fact measures the heterogeneity of a population, e.g., \( d = 0.5 \) implies that on average there are 50% different entries between two codes.

Topology: In SIRN, each code is represented by a node in the network, which describes the interactions among them. If there is an arc from node \( i \) to node \( j \), it indicates that node \( i \) receives information from node \( j \). At each time, a new network, called the topology of SIRN, is generated from the directed version of ER model with \( p = d \) (see Section 2). Fig. 3 shows a possible topology of SIRN which is directed and is not connected.

Neighborhood: For code \( i \), its neighborhood \( \mathcal{N}_i \) is defined to be the set of the codes from whom it receives information. In addition, a code is always included in its neighborhood\(^1\).

\(^1\)The term “neighborhood” defined here may not be the same as in some texts on graph theory.
Figure 3. The network describing the interactions among codes.

Figure 4. Demonstration of neighboring walks. Suppose the diversity of the population $d = 0.25$. For code $1$, with fitness $f_1 = 1$, it receives information from code 2 and 3, as is indicated by the topology. Thus the neighborhood of code 1, $\mathcal{N}_1 = \{1, 2, 3\}$, where $\mathcal{E}_1 = 3$. According to Eq. (2), $\Pr(c_{1,1} = 1) = 0.25$ and $\Pr(c_{1,1} = 0) = 0.75$, where $c_{1,1}$ denotes the first entry of code 1.

**Elite:** In this note, there are two kinds of elitists. For code $i$, the elite in its neighborhood $\mathcal{E}_i$ is the code with the highest fitness in $\mathcal{N}_i$, while the elite of the population, denoted by $\mathcal{E}$, is defined to be the code with the highest fitness in the population. If there are multiple codes with the highest fitness, randomly choose one.

**Neighboring walk:** For code $i$, the neighboring walk of its $k$th entry is defined as:

$$c_{i,k} = \begin{cases} \ c_{\mathcal{E}_i,k} & \text{with probability } d; \\ \ c_{i,k} & \text{with probability } 1 - d, \end{cases}$$

where $d$ is the diversity of the population. Note when the diversity $d$ decreases, the neighboring walk probability decreases. Fig 4 describes the neighboring walk of the first entry of a code.

**Random walk:** For code $i$, the random walk of its $k$th entry is defined as:

$$c_{i,k} = \begin{cases} \ 1 - c_{i,k} & \text{with probability } 1 - d; \\ \ c_{i,k} & \text{with probability } d. \end{cases}$$

### 3.2. Algorithm

In the following, the SIRN algorithm is formally defined.

**Step 1 Initialization:** A population is created and each code in the population is evaluated by the fitness function and is given a score, the fitness.

**Step 2 Elitist preservation:** Preserve the elite of the population.
Step 3 Regrouping: Generate a topology from the directed version of ER model.
Step 4 Walk: \( \forall i \in \{1, \ldots, n\}, \) if \( i = E_i, \forall j \in \{1 \ldots l\}, \) do random walk for \( c_{i,j}; \)
otherwise do neighboring walk for \( c_{i,j}. \)

If the stopping criteria are satisfied, terminate; else go to Step 3.2. The stopping
criteria can be specified like: the algorithm reaches a specified number of genera-
tions, the best fitness value in the current generation is less than or equal to a
specified value, for example.

3.3. The analysis of SIRN. Next it is proved that SIRN is able to find optimized
solutions almost surely. For convenience, the set of optimized solutions is referred
to as \( \mathcal{O}. \) For code \( x, \) the following function is given:

\[
\alpha(x) = \min_{x^* \in \mathcal{O}} ||x - x^*||,
\]

which measures the distance between \( x \) and \( \mathcal{O}. \) Associate the random sequence of
disagreement \( \{\alpha_k\}, \) which is defined by

\[
\alpha_k = \min_{x^* \in \mathcal{O}} ||X_k - x^*|| \geq 0,
\]

where \( X_k \) is a random variable of the elite in population at the \( k^{th} \) iteration. The
random sequence \( \{\alpha_k\} \) represents the trajectory of distance between the elite and
the optimized set \( \mathcal{O}. \) Note that \( \alpha_k = 0 \) indicates that an optimized solution is
found.

Theorem 1. For SIRN, optimized solutions are found almost surely.

Proof. Thanks to the step of elitist preservation, \( \alpha_{k+1} \leq \alpha_k. \) Consider the quantity
\( E[\alpha_{k+1}|\alpha_k = x] \) with \( x > 0, \) where \( E[\alpha_{k+1}|\alpha_k = x] \) is the conditional expected value
of \( \alpha_{k+1} \) given \( \alpha_k = x. \) Then one has

\[
E[\alpha_{k+1}|\alpha_k = x] = \sum_{k=1}^{N} y \cdot Pr(\alpha_{k+1} = y|\alpha_k = x),
\]

where \( N \) denotes the number of possible values of \( \alpha_{k+1} \) and \( Pr(\cdot) \) is the probability
function. Since \( \alpha_{k+1} \leq \alpha_k, \) one has \( y \leq x. \) Moreover, due to random walk, the
elite is able to switch to any other code (including the optimized ones) with a
nonzero probability, which implies that there exists at least a \( y < x \) such that
\( Pr(\alpha_{k+1} = y|\alpha_k = x) > 0. \) Hence,

\[
E[\alpha_{k+1}|\alpha_k = x] < x.
\]

From [12, Theorem 1], the conclusion comes directly. \( \square \)

4. Experimental Results

In this section, SIRN is tested on some benchmark test functions and a Job-Shop
Scheduling Problem (JSSP) to show its validity.

4.1. De Jong’s function 1. The first test function is De Jong’s function 1. It is
also known as sphere model and is continuous, convex and unimodal. It is defined as

\[
f(x) = \sum_{i=1}^{5} x_i^2, \quad -5.12 \leq x_i \leq 5.12,
\]

with global minimum

\[
f(x) = 0, \quad x = [0, \cdots, 0].
\]
Fig. 5 visualizes De Jong’s function 1 with two variables, each scaling within \([-5.12, 5.12]\).

**Figure 5. Visualization of De Jong’s function 1.**

Fig. 6 gives the comparison among elitist Genetic Algorithm (GA) [11], SIRN with the population size \(n = 50\) and \(n = 100\) respectively. All algorithms are run for 100 times, where the number of successful runs, i.e., the runs which find the optimum, is recorded. From Fig. 6, it can be seen that all the three algorithms found optimums in every run.

**Figure 6. The number of successful runs in 100 realizations.**
Fig 7 demonstrates the average generation for finding optimums. Worth noting here that in terms of generations, GA costs the most 35, while SIRN with population $n = 100$ costs the least 21.1.

4.2. Rosenbrock’s function. The second test function is Rosenbrock’s function\cite{10}, which is a classic optimization problem and is also known as De Jong’s function 2. The global optimum is inside a long, narrow, and parabolic shaped flat valley (See Fig 8). To find the valley is not difficult, however convergence to global optimum is difficult and hence this problem has been repeatedly used in assessing the performance of optimization algorithms. The function is described by

\begin{equation}
    f(x) = \sum_{i=1}^{2} \left( 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right)
\end{equation}

with global minimum

\begin{equation}
    f(x) = 0, \quad x = [1, \cdots, 1],
\end{equation}

where $-2.048 \leq x_i \leq 2.048$.

In this example, all algorithms are run for 100 times, while the number of successful runs is recorded in Fig 9. It is found that in this case GA is unable to find the optimum and the number of successful runs for SIRN with $n = 50$ and $n = 100$ was 47 and 84 respectively, which is much better than GA. Moreover, a key feature of an efficient and reliable optimization algorithm is the ability to maintain swarm diversity within a population of solutions. By examining Fig 10 one observes that the diversity of SIRN is kept much better than GA.

4.3. Job-Shop Scheduling Problem (JSSP). The Job-Shop Scheduling Problem is introduced in the sequel. An instance of the Job-Shop Scheduling Problem consists of a set of $n$ jobs and $m$ machines, where job $i$ consists of a sequence of $l_i$ activities. The $j$th activity of job $i$ has a start time $s_{i,j}$ and duration $d_{i,j}$ and requires a single machine for its entire duration. The machine used by the $j$th activity of job $i$ is denoted by $M_{i,j}$. An activity must be scheduled before every activity following it in its job. If two activities require the same machine, they cannot be scheduled at the same time. The objective is to find a schedule that
Figure 8. Visualization of Rosenbrock’s function.

Figure 9. The number of successful runs in 100 realizations.

Figure 10. The diversity of the three algorithms.
minimizes the overall completion time of all the activities. In its general form, it is NP-complete, meaning that there is probably no efficient procedure for exactly finding shortest schedules for arbitrary instances of the problem. The problem is described mathematically as follows:

\[
\text{objective : } \min_{s_{i,j}} \{ \max_{i} \{s_{i,l} + d_{i,l} \} \}
\]

\[
\text{s.t. } \begin{cases} 
 s_{i,k} - s_{i,k-1} \geq d_{i,k-1} & i \in \{1, \cdots, n \} \text{ and } 2 \leq k \leq l_i \\
 s_{i,j} - s_{p,q} \geq d_{p,q} & M_{i,j} = M_{p,q} \text{ and } s_{i,j} > s_{p,q}
\end{cases}
\]

Suppose an instance of the Job-Shop Scheduling Problem consists of a set of 3 jobs and 4 machines, which are listed in Table 1. For instance, Table(i, j)=(A,5), indicates that $M_{i,j} = A$ and $d_{i,j} = 5$, where Table(i, j) denotes the entry of the $i$th row and the $j$th column in Table 1.

In this case, there are 16 activities whose starting times $s_{i,j}$ need to be optimized, where $s_{i,j}$ denotes the starting time of the $j$th activity of job $i$. The fitness function is defined by Eq.(12), where the constraints are managed by punishment functions, i.e., if a constraint is not satisfied, the fitness $f$ is changed to $f/r \ (r > 1)$. According to Eq.(13), there are 13 constraints on jobs and 27 constraints on machines.

In the simulation, the population size $n = 50$. A starting time of an activity is encoded as a binary string of length 5. Thus, the length of a code $l = 16 \times 5 = 80$. The algorithm terminates if the max iterations, 300, is met. For this problem, SIRN is run for 10 times and the result is recorded in Table 2. It is found that all the ten runs are able to find the solutions satisfying all the 40 constraints while the optimum is 29 (Row 2 in Table 2).
5. Conclusions

This paper presents SIRN, a simple optimization algorithm using the swarm intelligence on random networks, which is motivated by recent studies on complex networks\cite{5, 6, 7}. The algorithm is written in a few lines of codes, and requires a few parameters. It seems that SIRN is effective in optimizing nonlinear functions and JSSP problem which is NP complete. It borrows the idea from the theory of random networks which is viewed as being nontrivial in social evolution. The network is derived from a directed version of ER model. Then by local interactions, the wisdom of the crowd emerges, namely it is able to find the optimum.

References


Mr. Fei Chen, is a Ph.D student in the Department of Automation, Nankai University. His current research interests include the areas of networked systems, nonlinear control systems, and optimization, with applications to unmanned aerial vehicles and biological networks.

Prof. Zengqiang Chen, received the B.S. degree in mathematics, M.S. degree and Ph.D. degrees in control theory from the Nankai University in China, in 1987, 1990, and 1997, respectively. He has been working at the College of Information Technology Sciences of Nankai University, where he is currently a professor. His main areas of research are complex networks, adaptive control, intelligent predictive control, and chaos system.

Dr. Linying Xiang, received the Ph.D. degree in control theory and control engineering from Nankai University in 2008. Currently, she is a postdoctoral researcher in the Department of Media and Communication, City University
of Hong Kong. Her present research interests include modelling, analysis and control of complex networks.

**Dr. Zhongxin Liu (1975)**, received his Ph.D. degree in Control Theory and Control Engineering from the department of Automation, Nankai University, P.R. China, June, 2002. Since 2004, he has been with the Predictive Control group in the department of Automation, Nankai University, as an assistant professor. His current research interests include control of nonlinear system, complex dynamical networks and multi-agent system.

**Prof. Zhuzhi Yuan**, received the B.S. degree in mathematics from the Nankai University, Tianjin, China, in 1962. From 1962 to 1984, he was an assistant professor at Department of Mathematics of Nankai University. He had been an associate professor from 1984 to 1989, and a professor since 1990 at the Department of Automation in Nankai University. His main areas of research are adaptive control, predictive control, and nonlinear control.

Department of Automation, Nankai University, Tianjin 300071, P. R. China

*E-mail: chenzq@nankai.edu.cn (Zengqiang Chen)*