Proceedings of the 38th Chinese Control Conference
July 27-30, 2019, Guangzhou, China

A Decentralized Continuous Estimation of Distribution Algorithm for Networked Systems

Aizhe Bian[1], Xi Chen[1]

1. Center for Intelligent and Networked Systems, Department of Automation, Tsinghua University, Beijing 100190, P. R. China
E-mail: baz16@mails.tsinghua.edu.cn,bjchenxi@tsinghua.edu.cn

Abstract: This paper proposes a decentralized continuous estimation of distribution algorithm to solve optimization problems in large scale networked systems which consist of many subsystems. The decentralized continuous estimation of distribution algorithm makes each subsystem cooperate with its neighbors to find good global solutions. Numerical examples illustrate the effectiveness of the algorithm.

Key Words: Decentralized systems, Networked systems, Optimization algorithms

1 Introduction

Over the decades, large scale networked systems such as sensor networks, intelligent building systems, electrical power grids, etc. are playing important roles in the society. Generally, networked systems consist many subsystems and these subsystems are physically dispersed. So we can view them as decentralized systems. Since networked system has expanded in size and extended in scale, the computational complexity has increased greatly. Centralized algorithms are becoming less efficient in practice. Moreover, centralized algorithms are sensitive to the change of systems' topologies. Due to the development of sensing, computation and communication technologies, the decentralized algorithms for optimization have developed rapidly in recent decades.

A large scale networked system can be viewed as a decentralized system. A decentralized system is combination of multiple subsystems. Each subsystem has a local objective, local decision variables and constraints. These subsystems are dispersed. Each subsystem can affect its neighbors. That is, the decision variables of a subsystem can also affect its neighbors' objective and constrains. The objective is to minimize the summation of local objectives of all subsystems. This problem is motivated by the problems in scheduling of industrial production systems[1], flow control of electricity market[2] and shared resource management[3], etc.

Decentralized methods have been proposed since the 1980s[4]. Bertsekas and Tsitsiklis proposed many iterative algorithms to deal with decentralized problems and present the conditions of convergence of these methods[4]. Inspired by "local autonomy and communication with neighbors", many researchers proposed synchronous decentralized algorithms[5]. In 2002, Inalhan et al. proposed a new decentralized algorithm to optimize a problem and prove its convergence under some assumptions[6]. In 2009, Yang et al. proposed a decentralized optimization algorithm to optimize the multiagent system-based watershed management[7]. In 2010, Loureiro et al. present an approach for managing shared resource pools in a decentralized, adaptive and optimal way[8]. In 2015, Hu and Chen proposed a decentralized ordinal optimization for optimization problems in networked system and apply adaptive learning in the method to narrow down the search space[9].

Apart from synchronous algorithms, in 2008, Huang et al. present an asynchronous decentralized algorithm for interconnected electricity markets[2].

Estimation of distribution algorithms (EDAs) are a series of evolution algorithms based on statistics[10–12]. The key point of EDAs is to estimate the distribution of good solutions selected from the generation and then generate the next generation by sampling based on the distribution. EDAs are population based algorithms that discard some individuals each generation and replace them using the statistical properties of highly-fit individuals[10].In 2007, Gallagher[13] proposed a continuous estimation of distribution algorithm called UMDA$_c^G$. This method use continuous probability instead of discrete probability to create the next generation.

This paper focuses on the decentralized continuous EDA to solve optimization problems for networked systems. We propose a new decentralized algorithm. The algorithm makes each subsystem optimize its own objective function independently based on local and neighbors' information. That is, each subsystems has a CPU to solve its problems in parallel and finds the global optimum. Therefore, this algorithm can solve more complex problems in a shorter time than a centralized method.

The rest of this paper is organized as follows. Section 2 describes the decentralized optimization problem. In Section 3, we discuss the decentralized algorithms. Section 4 reports the experimental results and provides analysis. Finally, we conclude this work with the important findings and anticipated future research directions.

2 Decentralized Problems

We hope to solve optimization problems for decentralized systems. For a decentralized system, all subsystems are physically distributed and connected with their neighbors. We can use a graph $G(V, E)$, where V is the set of all vertexes and E is the set of all edges, to formulate the decentralized system. In graph $G(V, E)$, each vertex represents a subsystem and each edge represents a connection between two subsystems. Since all subsystems are connected directly or indirectly, the graph $G(V, E)$ is a connected graph. In addition, any two neighboring subsystems can share information with each other, the graph $G(V, E)$ is an undirected graph. In the graph $G(V, E)$, we can define $B(i)$ as the set

This work is supported by National Key Research and Development Project of China (No.20171231799).

of neighbors of subsystem i:

$$B(i) = \{j \in V : (i,j) \in E\} \tag{1}$$

Since $B(i)$ doesn't contain the vertex i, we define $\widetilde{B}(i)$:

$$\widetilde{B}(i) = B(i) \cup \{i\} \tag{2}$$

Without the loss of generality, we consider a network system which has n subsystems. According to our description, these subsystems have their own decision variables and share them with their neighbors. We denote the local decision variables of subsystem i as a vector $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D_i})^T$. The decision variables of the whole system can be denoted as $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$. Each subsystem has a local objective functionf_i and constrains.

In a decentralize system, each subsystem can affect its neighbors. That is to say, the decision variables of a subsystem can also affect its neighbors' objective function and constrains. Therefor, we define a new operator $\oplus$ for decision variables $\mathbf{x}_i$ and $\mathbf{x}_j$:

$$\mathbf{x}_i \oplus \mathbf{x}_j = (x_{i,1}, x_{i,2}, \ldots, x_{i,D_i}, x_{j,1}, x_{j,2}, \ldots, x_{j,D_j})^T \tag{3}$$

This equation represents the recombination of two vectors. So the decision variables of objective function f_i and constrains are:

$$\widetilde{\mathbf{x}}_i = \oplus_{j \in \widetilde{B}(j)} \mathbf{x}_j \tag{4}$$

Considering the optimization problem of a decentralized system, the goal is to minimize the objective function which is the sum of the objective function of all subsystems.

$$\begin{aligned} \min_{\mathbf{x}} F(\mathbf{x}) &= \sum_{\mathbf{x}} F(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = \sum_{i=1}^{n} f_i(\widetilde{\mathbf{x}}_i) \\ &= f_1(\widetilde{\mathbf{x}}_1) + f_2(\widetilde{\mathbf{x}}_2) + \cdots + f_n(\widetilde{\mathbf{x}}_n) \end{aligned} \tag{5}$$

For a decentralize system, each subsystem can only decide the value of its local decision variables and calculate its own objective function. That is, all subsystems don't know the others' objective functions and constrains. According to the equation (5), the decision variable $\mathbf{x}_i$ can only affect its neighbors' objective functions and constrains. Since each subsystem can share information with its neighbors, we can optimize equation (5) through decentralized algorithms.

3 Decentralized Continuous Estimation of Distribution Algorithms

For a decentralized system, each subsystem minimizes its objective function independently. We hope same algorithm can be applied to all subsystems. In this situation, the algorithm should be widely applied in different problems for the purpose of generalization ability. So we propose a decentralized continuous estimation of distribution algorithm here.

3.1 Estimation of Distribution Algorithms

Estimation of distribution algorithms (EDAs) optimize a function by keeping track of the statistics of the population of candidate solutions. Since the statistics of the population are maintained, the actual population itself does not need to be maintained from one generation to the next. A population is created at each generation from the last generation's population statistics, and then the statistics of the most fit individuals in the population are computed. Finally, a new population is created by using the statistics of the most fit individuals. This progress repeats from one generation to the next. So EDAs are population-based algorithms that discard at least part of the population each generation and replace it using the statistical properties of highly-fit individuals. EDAs differ from most evolution algorithms (EAs) in that they typically do not include recombination. EDAs are also called probabilistic model-building genetic algorithms(PMBGAs), and iterated density estimation algorithms(IDEAs)[10].

In an EDA there are neither crossover nor mutation operators. The framework of an EDA is shown as follows.

Algorithm 1 Framework of an EDA

1: Initialize a population of candidate solutions, population size is N
2: **while** not termination criterion **do**
3: Compute the objective function value of each individual;
4: Select M individuals from the population based on some rules;
5: Estimate the distribution of good solutions based on the M individuals;
6: Sample from the distribution to generate the new generation;
7: **end while**
8: Choose the solution with least function value.

For some real-world problems, the function value is measured rather than calculated. What we get is a set of data and we can't use the gradient method to solve the optimization problem. An EDA doesn't need the gradient information. It can be more widely applied.

EDAs have different ways for estimating the distribution. Here we use the continuous distribution. Compared with EDAs for discrete-domain problems, EDAs for continuous-domain can search for more solutions. What's more, we need fewer parameters to describe the distribution in continuous EDAs.

3.2 Procedure of Decentralized Continuous EDA

We design a DC-EDA(Decentralized Continuous Estimation of Distribution Algorithm) based on EDAs. A subsystem, say subsystem i , can follow the following steps.

Step 1: Set the initial iteration number as 0, and set the maximum iteration number.

Step 2: Initialize a population of candidate solutions, population size is N.

Step 3: Pass its distribution to its neighbors and get neighbors' distributions.

Step 4: Generate the decision variables from neighbors' distributions.

Step 5: Calculate the objective function value of all individuals.

Step 6: Select K individuals according to their performance($M < K < N$).

Step 7: For each selected individuals, find the most similar k individuals from its neighbors. Then calculate the average value of these k individuals as the influence on neighbors.

Step 8: Calculate the cost of each individual. The cost contains the objective function value and the influence on neighbors.

Step 9: Select M individuals based on cost from the K individuals.

Step 10: Calculate the distribution of the M individuals.

Step 11: Sample from the distribution to generate the next generation.

Step 12: If the termination criterion is satisfied, go to Step 13. Otherwise, go to Step 3.

Step 13: Obtain the solutions.

Subsystems need to cooperate with each other to adjust the value of local decision variable to minimize the objective function. The objective function is loss function in this paper. So we use cost instead of loss to select the good individuals in Step 8.

$$cost = loss + influence \tag{6}$$

Then we need to calculate the influence in Step 7. We can change the form of the whole loss function as follows:

$$\begin{aligned}\min_{\mathbf{x}_i} f(\mathbf{x}) &= \sum_{j\in\widetilde{B}(i)} f_j(\widetilde{\mathbf{x}}_j) + \sum_{j\notin\widetilde{B}(i)} f_j(\widetilde{\mathbf{x}}_j) \\ &= f_i(\widetilde{\mathbf{x}}_i) + \sum_{j\in B(i)} f_j(\widetilde{\mathbf{x}}_j) + \sum_{j\notin\widetilde{B}(i)} f_j(\widetilde{\mathbf{x}}_j)\end{aligned} \tag{7}$$

The second formula $\sum_{j\in B(i)} f_j(\widetilde{\mathbf{x}}_j)$ represents the influence on neighbors. But we can't calculate it directly. For each individual in subsystem i, we can find part of decision variables x_{ji} that belongs to subsystem j. Then we can find k individuals closest to x_{ji} in subsystem j. We use $\mathbf{x}_{\hat{j}}$ to represent these k individuals. Then we can use the average loss values of these individuals to represent the influence as shown in (5).

$$influence(\mathbf{x}_i) = \sum_{j\in B(i)} (\sum_{k} f_j(\mathbf{x}_{\hat{j}})/k) \tag{8}$$

Now we discuss the computational complexity of the decentralized continuous EDA. We ignore the cost of data exchange between neighbors because it depends on the communication network. For each subsystem, we need to calculate N loss function values in an iteration. Compared with centralized EDA, we needn't calculate extra loss function values in an iteration. However, we need find some individuals and calculate the average of their loss function values in an iteration. These operations need much less computational resource than calculating loss functions. Meanwhile, the information transfer between neighbors also need extra operations. So the computational complexity of a decentralized continuous EDA and a centralized continuous EDA is similar in an iteration if they have the same parameters. Considering that the computing resource is distributed on each subsystem, the decentralized EDA can perform better.

The decentralized continuous EDA also has the advantage of flexibility. That is, if the decentralized system loses a subsystem, we only need change the codes of its neighboring subsystems other than of the whole system.

4 Experimental Results

Decentralized systems have a variety of topologies. Different topologies may affect the effectiveness of decentralized algorithms. Therefore, we apply our algorithm to different systems.

4.1 Example 1: topology of chain

The topology of chain is shown as follows:

Fig. 1: 4 subsystems with a topology of chain

In the problem statements below we use o_i to refer to a random offset and we use M to refer to an random rotation matrix. Therefore, the optimization problems are more complex and representative.The loss functions are as follows:
Subsystem 1: the Rastrigin function;

$$\begin{aligned}\min f_1(\mathbf{x}) =& 10\times 4 + z_{11}^2 + z_{12}^2 + z_{13}^2 + z_{23}^2 \\ &- 10\times(\cos(2\pi z_{11}) + \cos(2\pi z_{12}) \\ &+ \cos(2\pi z_{13}) + \cos(2\pi z_{23}))\end{aligned} \tag{9}$$

where $z_i = Mx_i - o_i$.
Subsystem 2: the quartic function;

$$\min f_2(\mathbf{x}) = z_{21}^4 + 2z_{22}^4 + 3z_{23}^4 + 4z_{13}^4 + 5z_{33}^4 \tag{10}$$

where $z_i = Mx_i - o_i$.
Subsystem 3: the Griewank function;

$$\begin{aligned}\min f_3(\mathbf{x}) =& 1 + (z_{31}^2 + z_{32}^2 + z_{33}^2 + z_{23}^2 + z_{43}^2)/4000 \\ &- [\cos(z_{31})\cos(z_{32}/\sqrt{2})\cos(z_{33}/\sqrt{3}) \\ &\cos(z_{23}/\sqrt{4})\cos(z_{43}/\sqrt{5})]\end{aligned} \tag{11}$$

where $z_i = Mx_i - o_i$.
Subsystem 4: the Schwefel absolute function;

$$\begin{aligned}\min f_4(\mathbf{x}) =& |z_{41}| + |z_{42}| + |z_{43}| + |z_{33}| \\ &+ |z_{41}z_{42}z_{43}z_{33}|\end{aligned} \tag{12}$$

where $z_i = Mx_i - o_i$.

The loss functions of subsystem 1, subsystem 2 and subsystem 4 are multimodal. The loss function f_4 is nonderivable. The overall loss of this problem $F(\mathbf{x}) = \sum_{i=1}^{4} f_i(\mathbf{x})$ is a multimodal function and nonderivable. Since there are

random offset and random rotation matrix, we don't know the optimal solution here. So we just compare the performance of the centralized EDA and our decentralized continuous EDA.

The overall loss function $F(\mathbf{x})$ is a 12-dimensional function. So we can set population size $N = 1000$ as a rule of thumb. And we can set elitism parameter $E = 2$, which means that we always keep the best two individuals from one generation to the next. The number of selected individuals to estimate the distribution is $M = 0.1N$ according to the experience. For decentralized EDA, the number of first selection is $2M$. We set $k = N/(2M)$. We perform 40 Mote Carlo simulations with the same random offset and random rotation matrix.

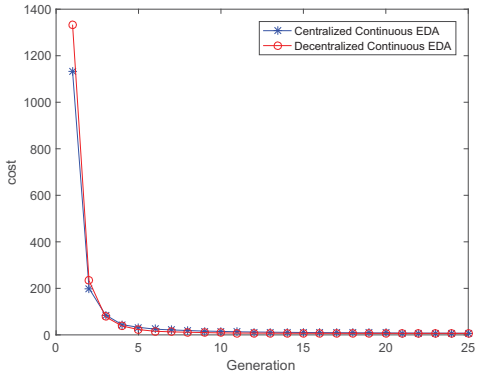

Fig. 2: Example 1: Results for the system

Figure 2 shows the best individual at each generation, averaged over 40 Monte Carlo simulations. We can find that these two algorithms perform nearly the same. More details of the solutions are shown in the following figure and table.

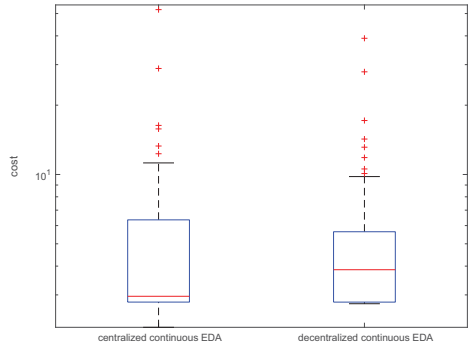

Fig. 3: Boxplot of Centralized Continuous EDA and Decentralized Continuous EDA

Figure 3 is the boxplot of the 40 Monte Carlo simulation results for these two algorithms. Each box shows the middle 50% of the set of results for an algorithm. The line inside each box shows the median result. The lines above and below each box show the maximum and minimum results.

From Figure 3 and Table 1 we can find that the two algorithms perform similarly. The best solution of the centralized EDA is better than the DC-EDA. The median solution, the average solution and the standard deviation of the decentralized EDA is better. The standard deviation shows that the DC-EDA is more robust than the centralized EDA.

Table 1: Comparison between centralized EDA and Decentralized EDA

Algorithm	Best solution	Median solution	Average solution	Standard deviation
EDA	2.1693	2.9584	6.7889	9.1104
DC-EDA	3.8584	1.9076	6.5367	7.3564

Then we consider different situations. In each Monte Carlo simulation, we use different random offsets and random rotation matrix. Then we compare the solutions between these two algorithms. The following figure shows the difference between these two algorithms in each Monte Carlo simulation.

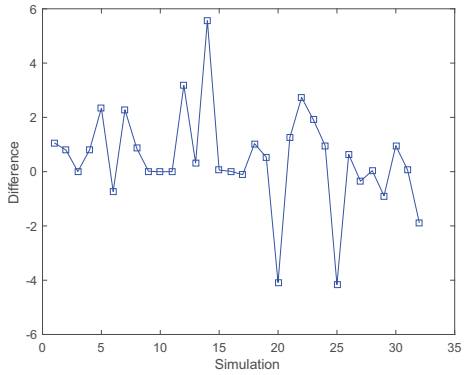

Fig. 4: Difference between these two algorithms

From Figure 4 we can find that the performance of these two algorithms are similar in different problems. We can conclude that the Dc-EDA can achieve the effectiveness of centralized continuous EDA in the topology of chain. As the DC-EDA makes all subsystems optimize their local problems in parallel, the DC-EDA runs faster than the centralized continuous EDA.

4.2 Example 2: topology of grid

The topology is shown in Figure 5.

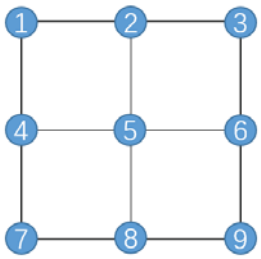

Fig. 5: 9 subsystems with a topology of grid

This topology is more complex. The dimension of decision variables in subsystem 1,3,5,7 and 9 is two. The dimension in subsystem 2,4,6 and 8 is one. In this problem, we use 7 kinds of benchmark functions as follows:

Subsystem 1: the Ackley function;

$$\begin{aligned}\min f_1(\mathbf{x}) =& 20 + e - 20\exp[-0.2(x_{11}^2 + x_{12}^2 + x_2^2 + x_4^2)/4] \\ & - \exp\{[\cos(2\pi x_{11}) + \cos(2\pi x_{12}) \\ & + \cos(2\pi x_2) + \cos(2\pi x_4)]/4\}\end{aligned} \tag{13}$$

Subsystem 2: the tenth power function;

$$\min f_2(\mathbf{x}) = x_2^{10} + x_{12}^{10} + x_{32}^{10} + x_{52}^{10} \tag{14}$$

Subsystem 3: the Griewank function;

$$\begin{aligned}\min f_3(\mathbf{x}) =& 1 + (x_{31}^2 + x_{32}^2 + x_2^2 + x_6^2)/4000 \\ & - [\cos(x_{31})\cos(x_{32}/\sqrt{(2)}) \\ & \cos(x_2/\sqrt{3})\cos(x_6/\sqrt{4})]\end{aligned} \tag{15}$$

Subsystem 4: the Ackley function;

$$\begin{aligned}\min f_4(\mathbf{x}) =& 20 + e - 20\exp[-0.2(x_4^2 + x_{12}^2 + x_{52}^2 + x_{72}^2)/4] \\ & - \exp\{[\cos(2\pi x_4) + \cos(2\pi x_{12}) \\ & + \cos(2\pi x_{52}) + \cos(2\pi x_{72})]/4\}\end{aligned} \tag{16}$$

Subsystem 5: the sphere function;

$$\min f_5(\mathbf{x}) = x_{51}^2 + x_{52}^2 + x_2^2 + x_4^2 + x_6^2 + x_8^2 \tag{17}$$

Subsystem 6: the Rastrigin function;

$$\begin{aligned}\min f_6(\mathbf{x}) =& 10 \times 4 + x_6^2 + x_{32}^2 + x_{52}^2 + x_{92}^2 \\ & - 10 \times (\cos(2\pi x_5) + \cos(2\pi x_{32}) \\ & + \cos(2\pi x_{52}) + \cos(2\pi x_{92}))\end{aligned} \tag{18}$$

Subsystem 7: the quartic function;

$$\min f_7(\mathbf{x}) = x_{71}^4 + 2x_{72}^4 + 3x_4^4 + 4x_8^4 \tag{19}$$

Subsystem 8: the Griewank function;

$$\begin{aligned}\min f_8(\mathbf{x}) =& 1 + (x_8^2 + x_{52}^2 + x_{72}^2 + x_{92}^2)/4000 \\ & - [\cos(x_8)\cos(x_{52}/\sqrt{2}) \\ & \cos(x_{72}/\sqrt{3})\cos(x_{92}/\sqrt{4})]\end{aligned} \tag{20}$$

Subsystem 9: the Schwefel absolute function;

$$\begin{aligned}\min f_9(\mathbf{x}) =& |x_{91}| + |x_{92}| + |x_6| + |x_8| \\ & + |x_{91}x_{92}x_6x_8|\end{aligned} \tag{21}$$

The overall loss of this problem $F(\mathbf{x}) = \sum_{i=1}^{9} f_i(\mathbf{x})$ is a 14-dimensional function. This function is a multimodal function and nonderivable. We use a domain of $[-5, 5]$ for each dimension. For the system, the optimal solution is $\mathbf{x}^* = 0, F(\mathbf{x}^*) = 0$ without random offset and random rotation matrix. We set $N = 1000, E = 2, M = 0.1N$ and $k = N/(2M)$ like the former problem. The results are as follows.

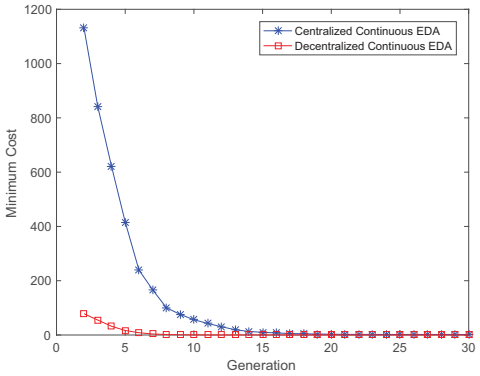

Fig. 6: Example 2: Results for the system

Figure 6 shows the best individual at each generation, averaged over 40 Monte Carlo simulations. From Figure 6, we can find that the DC-EDA converges faster than the centralized continuous EDA. More details of the solution and decision variables are shown in the following figure and tables.

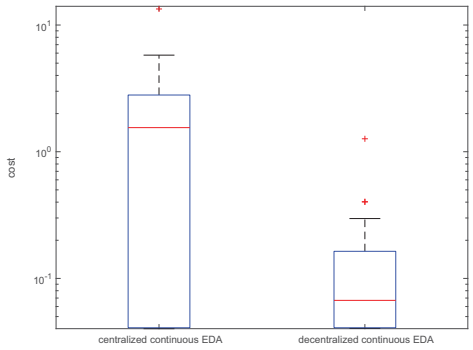

Fig. 7: Boxplot of 40 Monte Carlo Simulation Results for Centralized Continuous EDA and Decentralized Continuous EDA

Table 2: Comparison between centralized continuous EDA and DC-EDA

Algorithm	Best solution	Median solution	Average solution	Standard deviation
EDA	0.0401	1.5943	1.9134	2.5908
DC-EDA	0.0401	0.0671	0.1365	0.2065

From Figure 7 and Table 2 we can find that the decentralized EDA can find better solutions than the centralized EDA in this problem. The median solution, average solution and standard deviation of the decentralized EDA are smaller than the centralized EDA. The condition of decision variables is shown in the next table.

Table 3: Comparison of distance to optimal decision variables between centralized EDA and decentralized EDA

Algorithm	Least distance	Median distance	Average distance	Standard deviation
EDA	0.0183	0.8110	0.9194	1.6348
DC-EDA	0.0183	0.0248	0.1341	0.3862

The optimal decision variable is $\mathbf{0}$. We calculate the Euclidean distance between our decision variables and the optimum in each simulation. From Table 3 we can find that decision variables of the decentralized continuous EDA is closer to the optimal decision variables $\mathbf{0}$. This is consistent with previous results.

Compared to the previous example, we can find the performance of the DC-EDA is better here. We assume that it is related to searching ability. In this problem, there are 9 subsystems and the loss function is complex. Each subsystem has N independent candidate solutions. All subsystems share the distribution and some candidate solutions with neighbors. That means the whole system can search more candidate solutions in an iteration in the DC-EDA. So DC-EDA converges faster than centralized continuous EDA here. The searching ability increases with the expansion of scale. In addition, the information from neighbors make the candidate solutions away from local optimum. Subsystems in this example have at least 2 neighbors and subsystems in the previous example have at most 2 neighbors. More neighbors mean more information.

These two examples show that the performance of the DC-EDA is close to the centralized continuous EDA. In addition, the DC-EDA converges faster than the centralized continuous EDA.

5 Conclusions

In this paper, we develop the DC-EDA to deal with decentralized optimization problems for networked systems. The advantage of DC-EDA is that it can search more solutions in an iteration. Therefore, it can find better solutions in a shorter time. Meanwhile, it has the advantage of flexibility so that it can be widely applied to various problems in networked systems.

We observe that the DC-EDA can be combined with other evolution algorithms and gradient method to accelerate the iterative progress. This will be our future work.

References

[1] N. Shah, G. K. D. Saharidis, Z. Jia, and M. G. Ierapetritou, "Centralizeddecentralized optimization for refinery scheduling," *Computers & Chemical Engineering*, vol. 33, no. 12, pp. 2091–2105, 2009.

[2] A. Huang, S.-K. Joo, K.-B. Song, J.-H. Kim, and K. Lee, "Asynchronous decentralized method for interconnected electricity markets," *International Journal of Electrical Power & Energy Systems*, vol. 30, no. 4, pp. 283–290, 2008.

[3] T. Nowicki, M. S. Squillante, and W. W. Chai, *Fundamentals of dynamic decentralized optimization in autonomic computing systems*, 2005.

[4] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[5] F. Bullo, J. Cortés, and S. Martinez, *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009.

[6] G. Inalhan, D. M. Stipanovic, and C. J. Tomlin, "Decentralized optimization, with application to multiple aircraft coordination," in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 1. IEEE, 2002, pp. 1147–1155.

[7] Y.-C. E. Yang, X. Cai, and D. M. Stipanović, "A decentralized optimization algorithm for multiagent system–based watershed management," *Water resources research*, vol. 45, no. 8, 2009.

[8] E. Loureiro, P. Nixon, and S. Dobson, "Adaptive management of shared resource pools with decentralized optimization and epidemics," in *Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on*. IEEE, 2010, pp. 51–58.

[9] P. Hu and X. Chen, "Decentralized ordinal optimization (doo) for networked systems," in *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*. IEEE, 2015, pp. 787–792.

[10] D. Simon, *Evolutionary optimization algorithms*. John Wiley & Sons, 2013.

[11] P. Larrañaga and J. A. Lozano, *Estimation of distribution algorithms: A new tool for evolutionary computation*. Springer Science & Business Media, 2001, vol. 2.

[12] E. Bengoetxea, P. Larrañaga, I. Bloch, and A. Perchant, "Estimation of distribution algorithms: A new evolutionary computation approach for graph matching problems," in *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer, 2001, pp. 454–469.

[13] M. Gallagher, I. Wood, J. Keith, and G. Sofronov, "Bayesian inference in estimation of distribution algorithms," in *IEEE Congress on Evolutionary Computation*, 2007.