A stochastic search approach for the multidimensional largest empty sphere problem

Jong-Seok Lee, Taeg-Sang Cho, Jiye Lee, Myung-Kee Jang, Tae-Kwang Jang, Dongkyung Nam, and Cheol Hoon Park, *Senior Member, IEEE*

Abstract

This paper presents a novel approach to solve the largest empty sphere (LES) problem in the multidimensional space by using a popular stochastic search approach, evolutionary algorithm (EA). When a set of points are given in a space, the LES problem is to find a point from which the distance to the nearest point among the set is maximized. Conventionally, the LES problem can be solved by the use of the Voronoi diagram which is a useful data structure in the field of computational geometry. However, we have difficulty in constructing the Voronoi diagram in the high-dimensional space because the time and the storage complexities grow exponentially as the dimension of the problem becomes high. In this paper, an EA approach is used as an effective method of finding the solution of the LES problem in the multidimensional Euclidean space. Experimental results show that the proposed method successfully finds the solutions of the LES problems in various dimensional spaces and is efficient in terms of the execution time and the necessary memory in comparison with the Voronoi diagram approach.

Index Terms

largest empty sphere problem, evolutionary algorithm, Voronoi diagram.

This work was supported by grant No. R01-2003-000-10829-0 from the Basic Research Program of the Korea Science and Engineering Foundation.

A stochastic search approach for the multidimensional largest empty sphere problem

I. Introduction

The largest empty sphere (LES) problem is a frequently addressed proximity problem in the field of the computational geometry [1]. Given a set of points (sites) in a space, the objective of the problem is to find a point from which the distance to the nearest site is the longest. The practical importance of this problem arises in several areas including industrial engineering, geometrical information systems and operational research [2]. For example, suppose that you are given N points in the plane, representing cities. You should put a toxic waste dump as far from any of the cities as possible in order to minimize the influence of the toxic waste to the people living in the cities. In this case, the best place of the dump will be the point from which the distance to the nearest city is maximized.

The LES problem is stated formally as follows:

> *Largest Empty Sphere Problem*: Given a set S_p of N points $s^i, i = 1, ..., N$, in a bounded space S, find the point p in S from which the distance to the nearest point among the set is the longest, i.e.,
>
> $$\max_{p \in S} \min_{i} d(p, s^i), \tag{1}$$
>
> where $d(p, q)$ is the distance between two points p and q.

When we specify the space S as the two-dimensional Euclidean space, the problem becomes the *largest empty circle* problem [3] and the "toxic waste dump problem" mentioned above falls into this category. Also, the space S can be the surface of a (three-dimensional) sphere, which is the case that the cities are distributed over the entire earth.

The LES problem can be solved using the *Voronoi diagram* [4], which is a simple and powerful spatial data structure in computational geometry. Given a set S_p of N points in a space, the Voronoi diagram partitions the space into N convex polyhedral regions so that the Voronoi cell of each given point s^i contains the set of points which are closer to s^i than any other points in S_p. Once the Voronoi diagram is constructed, the center of the largest empty sphere can be obtained by finding the Voronoi vertex which maximizes the distance to the nearest point in S_p.

There are several algorithms for constructing Voronoi diagrams in the plane [3], [5]. With the help of these work, we can solve the LES problem in the plane effectively. However, utilizing the Voronoi diagram becomes difficult when the problem is high-dimensional. The time and the storage complexities constructing the Voronoi diagram in the plane are given by $O(N \log N)$ and $O(N)$, respectively, where N is the number of points, but they grow exponentially as the dimension of the space becomes higher. Moreover, while the Voronoi diagram in

the multidimensional hypercube has been studied extensively, there is little work for constructing the diagram in other geometric spaces such as the surface of a multidimensional hypersphere.

In this paper, we introduce a novel approach using the *evolutionary algorithm* (EA) [6], [7] to solve the LES problem in the multidimensional space. The EA is one of the most popular stochastic search algorithms and has been successfully applied in various kinds of problems such as combinatorial/real-valued optimization problems with differentiable/non-differentiable objectives. Especially, it has been a good method for the problems which are difficult to solve mathematically or analytically, and thus, is well-suited for the multidimensional LES problem. We design the two-stage searching process [8], [9] by using the *search space renormalization scheme* in which the search domain is reduced around the best solution so far after the global search is converged in some degree. This procedure enhances the fine-tuning capability of the EA and helps to obtain as accurate solutions as the Voronoi diagram method yields. We demonstrate the performance of the proposed approach in comparisons with that of the Voronoi diagram approach in multidimensional spaces.

In the following section, we discuss the method of solving the LES problem by using the Voronoi diagram. Section III describes how to solve the problem by using the EA. The simulation results are given in Section V, and finally, conclusion is made in Section VI.

II. Voronoi Diagram Approach

Let S_p the set of the given N points s^i ($i = 1, 2, ..., N$), as in the previous section. A Voronoi diagram is the partitioning of a space with the N points into N convex polyhedrons such that each polyhedron contains exactly one point and every point in a given polyhedron is closer to its central point than to any other. The cells are called *Voronoi polyhedrons* (*Voronoi polygons* in the case of the plane), the vertices of the diagram are *Voronoi vertices*, its line segments are *Voronoi edges* and the extreme rays (i.e., unbounded edges) are *Voronoi rays* as shown in Fig. 1. Each of the given N points belongs to a unique Voronoi polyhedron and, if a point p is contained in the Voronoi polyhedron associated with s^i, then s^i is the nearest neighbor of p.

The Voronoi diagram is used to solve the LES problem as follows [2]. Consider the function $f(p)$, the distance of point p to the nearest point among S_p. Within the Voronoi polyhedron which contains p and is associated with s^i, $f(p)$ is a unimodal function which has the minimum of zero at s^i, and the same applies for each polyhedron of the described partition. Thus, $f(p)$ attains its maximum at a vertex of one such polyhedron. Therefore, the candidates of the center of the largest empty sphere are the vertices of the Voronoi diagram and, among them, we choose the one having the largest radius. Fig. 1 shows an example of solving the largest empty circle problem using the Voronoi diagram.

Several algorithms for constructing the Voronoi diagrams in $\mathbb{R}^2$ have been developed in the field of computational geometry [3], [5]. It was proven that the Voronoi diagram in the plane can be computed in $O(N \log N)$ time and with $O(N)$ storage (actually the number of the vertices in the diagram is at most $2N - 5$, which comes from Euler's formula) [2], [3].

Difficulty for solving the LES problem by the Voronoi diagram in the high-dimensional space lies in the exponential growth of the complexity. In the D-dimensional Euclidean space, the Voronoi diagram of a set of N points is computed in $O(N \log N + N^{\lceil D/2 \rceil})$ optimal time [10] and its maximum combinatorial complexity

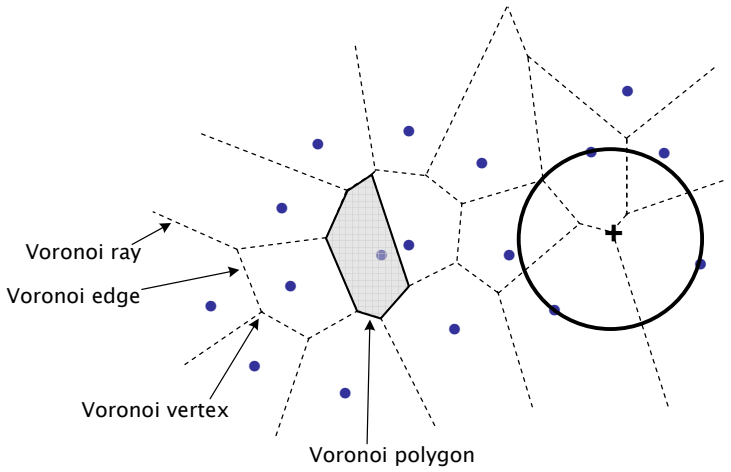

Fig. 1. A Voronoi diagram (dashed line) and the largest empty circle (its center is marked with +).

(the maximum number of vertices, edges, and so on, of the diagram) is $\Theta(N^{\lceil D/2 \rceil})$ [11]. In order to overcome this difficulty, we propose an EA approach for solving the LES problem in the high-dimensional space in the next section.

III. EA Approach

In our EA approach, we use the real-value encoding scheme and thus the coordinates of the solution is directly used as the parameter values of the chromosome. The fitness function of the solution x directly comes from Eq. (1):

$$C(x) = \min_i d_E(x, s^i), \tag{2}$$

where s^i is the i-th given point and $d_E(p, q)$ the Euclidean distance of two points p and q. Therefore, the objective of the problem is to maximize $C(x)$ with respect to x.

Fig. 2 shows an example of the fitness landscape of the parameter space obtained by Eq. (2) when the dimension is two and 100 points are chosen in $[0, 1]^2$ randomly. As shown in the figure, there are many local optima (maxima) having similar costs at the sharp peaks. Actually, the number of local optima is equal to the number of the Voronoi vertices.

In order to improve the fine-tuning capability of the EA, we adopt a two-stage searching process [8], [9] by using the search space renormalization scheme. This concept is illustrated in Fig. 3. In the first stage, the search domain is the entire parameter space as usual. After the criteria for terminating the first stage are satisfied, the search focuses on the reduced search space around the best solution found in the first stage. In this case, the parameter values of the chromosomes and the coordinates of the given points are renormalized (magnified) so that we get more precise solutions.

The overall EA procedure including the two-stage searching process is summarized as follows.

Procedure EA for the LES problem

STAGE 1

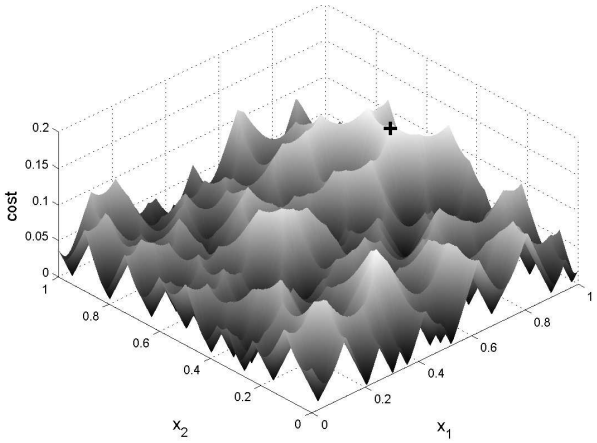

Fig. 2. Example of the landscape when 100 points are given in the plane. The global optimum is marked with +.

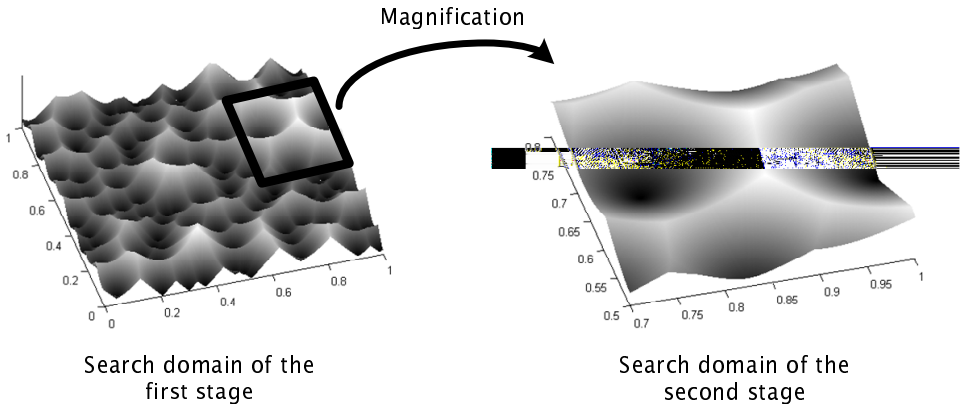

Fig. 3. Search space renormalization scheme.

Step 1. Initialize and evaluate μ chromosomes.

Step 2. Generate μ offsprings from μ parents by mutation. Evaluate them.

Step 3. Select μ chromosomes among 2μ candidates (μ parents + μ offsprings) for the next generation.

Step 4. If the termination conditions are satisfied, go to Step 5. Else, go to Step 2.

STAGE 2

Step 5. Reduce the search space around the best chromosome. Magnify the space. Move all the chromosomes into the reduced domain.

Step 6. Generate μ offsprings from μ parents by mutation. Evaluate them.

Step 7. Select μ chromosomes among 2μ candidates for the next generation.

Step 8. If the termination conditions are satisfied, stop. Else, go to Step 6.

We start the algorithm with μ chromosomes which are initialized by random number generation. Each chromosome becomes a parent chromosome and produces an offspring by mutation. Then, the parents and the generated offsprings consist of a genetic pool from which the population of the next generation is selected. This procedure continues and, if some conditions are satisfied, the first stage is terminated and the second stage starts. At the beginning of the second stage, the search space is reduced around the position of the best chromosome and the coordinates of the given points are multiplied by some constant to magnify the space. The chromosomes which are out of the new space are spread randomly within the new space. We repeat the procedure of generation of the offsprings and selection of the chromosomes as in the first stage. When some termination conditions are satisfied, the overall EA procedure is terminated.

We use the Gaussian mutation operator which perturbs each gene of a chromosome by the Gaussian distribution, i.e.,

$$x'_j = x_j + N(0, \sigma_k), \tag{3}$$

where x_j is the j-th parameter of a chromosome, σ_k the standard deviation of the Gaussian distribution at the k-th generation, and $N(0, \sigma_k)$ Gaussian distribution with a mean of zero and a standard deviation of σ_k. σ_k represents the degree of mutation at the k-th generation and gradually decreases as time goes:

$$\sigma_k = \frac{\sigma_0}{k}. \tag{4}$$

σ_0 should be large enough to render the chromosome have a chance to explore the entire domain sufficiently at the beginning of the algorithm.

After each parent produces an offspring by mutation, the parents and the offsprings compose the pool for the population of the next generation. We use the binary tournament scheme for selection; two chromosomes in the pool are chosen randomly and the one having the better cost is inserted in the population of the next generation; this is repeated μ times. We apply the best-keeping scheme in which the best chromosome in the pool is always kept in the next generation.

Each stage is terminated when the cost does not improve significantly during some generations. We judge that the improvement of the cost is saturated if the cost of the best chromosome is not improved by $R\%$ during k consecutive generations. R and k can be set to different values for each stage in order that we have the flexibility for the tuning of the algorithm.

IV. EXPERIMENTS

We test the proposed approach via computer simulations. We compare the EA approach with the Voronoi diagram approach. We make 30 problems with N data points ranging from 1000 to 5000 and D (dimension) ranging from two to seven. Each data point is randomly generated in $[0, 1]^D$. Both algorithms are implemented in C language. Especially for generating the Voronoi diagram, we use the Quickhull package [12]. We run the EA program five times with different initial random values. The results on a PC with a Pentium-4 3.2 GHz CPU and 2 GB RAM are reported in Table I and Fig. 4. The Voronoi diagram approach failed to solve the problems with more than 1000 points in $D = 7$ because the number of the Voronoi vertices are too much to be

stored. Hence, the results for these cases are omitted in the table and in the figure. For the 26 problems which the Voronoi diagram approach is successfully applied, the EA succeeded in finding the solutions for all trials.

We use the population of size 50 (i.e., $\mu = 50$) in our EA approach. The initial standard deviation for the Gaussian mutation scheme, σ_0, is set to 100, which is sufficiently large for exploring the entire domain. The first stage of the EA is terminated when the best cost is not improved by 1% during 3000 generations. This criterion was determined on the basis of our observation that the chromosomes sufficiently explore the parameter space until this criterion is met. The second stage is terminated when the improvement is less than 0.1% during 1000 generations since we can obtain sufficiently accurate solutions with this criterion in our experiments. The number of the maximum generation is set to be large enough that the algorithm converges sufficiently and is terminated by the stopping criteria.

Table I shows the errors of the costs and the positions of the solutions found by the EA in comparison with those of the exact solutions by the Voronoi diagram approach. E_c is the relative error of the cost by the EA to that of the Voronoi diagram method, i.e.,

$$E_c = \frac{C^V - C^E}{C^V} \times 100\%, \tag{5}$$

where C^V and C^E are the costs of the solutions found by the Voronoi diagram method and the EA, respectively. E_p is the absolute maximum error in the position of the solution, i.e.,

$$E_p = \max_j |x_j^V - x_j^E|, \tag{6}$$

where x_j^V and x_j^E are the j-th coordinates of the solutions found by the Voronoi diagram method and the EA, respectively.

The results in the table show that the EA successfully finds the solutions for all the 26 problems. The relative errors in costs are less than about 0.2% on average (0.3% in worst cases) and the errors in the parameter values are maximally the orders of 10^{-3} on average. We see that E_p becomes larger as the dimension becomes higher, which is due to the high complexity of the high-dimensional problems.

Fig. 4 compares the CPU time for executing the Voronoi diagram method and the EA method. For the EA, the average values over five trials are shown. When the dimension is low, the Voronoi diagram method is much quicker than the EA. However, the inversion begins to occur when the dimension is 5 and, when $D = 6$, the EA is much faster than the Voronoi diagram approach. The required CPU time for both methods increases as the number of points increases and the dimension becomes larger, but the rate of the growth in the EA is much less than that in the Voronoi diagram method. For example, when the dimension becomes higher from two to six with $N = 5000$, the time for the Voronoi diagram method increases by the factor of about 1690 (from 0.6 to 1014 seconds) while the CPU time for the EA increases by the factor of just 1.6 (from 38 to 61 seconds).

Memory requirement is another important difference between the Voronoi diagram method and the EA. The number of vertices of the Voronoi diagram exponentially increases, and so does the memory occupation for saving the vertices. However, we need to save only $2 \times D \times \mu$ for the parents and the offsprings in the EA. When $D = 6$ and $N = 5000$, for example, it was necessary to save about 4.5×10^6 vertices in the Voronoi diagram while the EA has only 600 parameters. For the problems with more than 1000 points in the dimensions

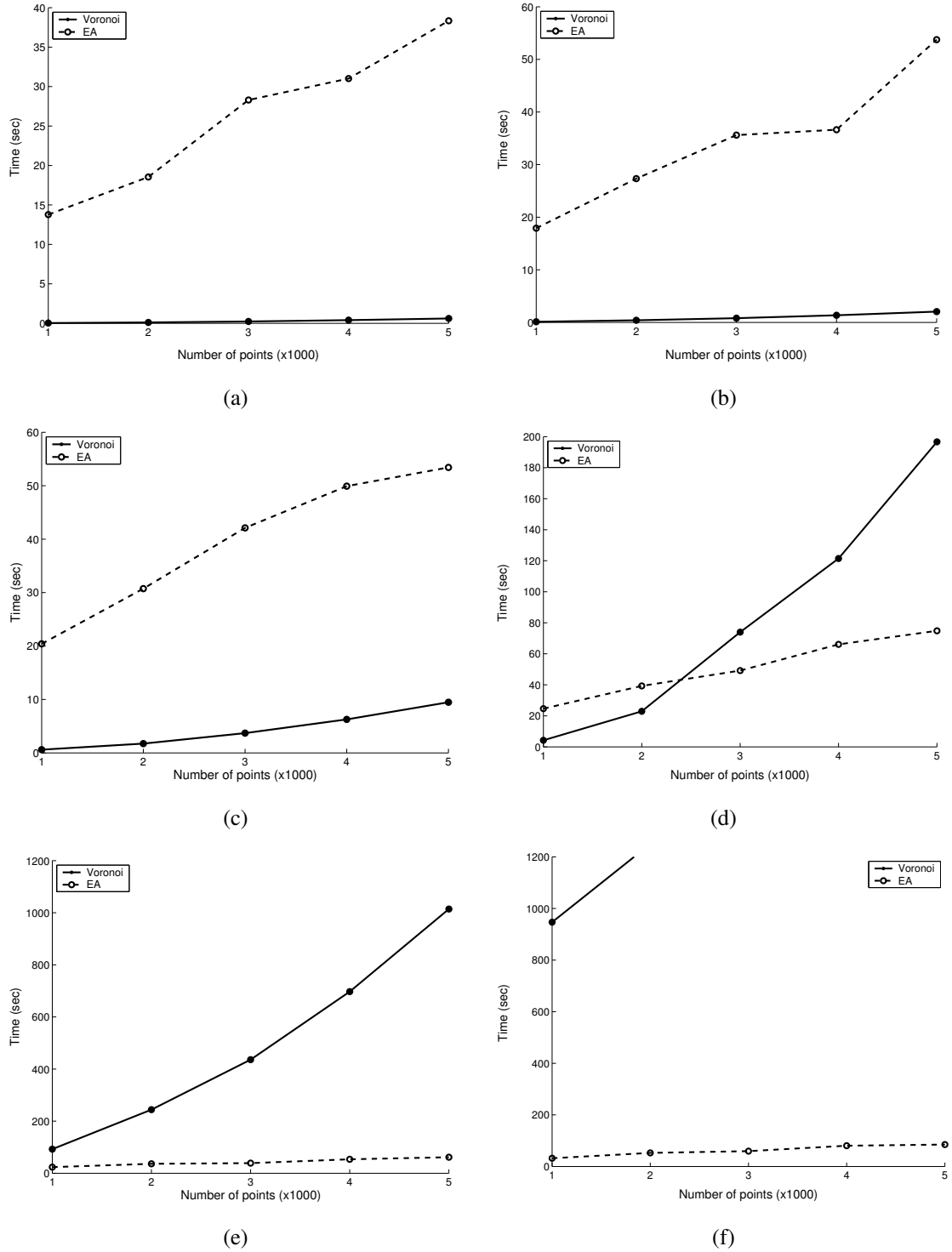

Fig. 4. Comparison of the CPU time of the Voronoi diagram approach and the proposed EA approach when (a) D=2, (b) D=3, (c) D=4, (d) D=5, (e) D=6 and (f) D=7.

TABLE I

PERFORMANCE OF THE PROPOSED APPROACH IN THE VIEWPOINT OF THE SOLUTION QUALITY. AVERAGE, MAXIMUM AND MINIMUM VALUES FOR FIVE TRIALS ARE SHOWN FOR EACH PERFORMANCE MEASURE.

Dim	#Points	$E_c(\%)$			E_p		
		Avg	Max	Min	Avg	Max	Min
2	1000	0.022	0.052	0.000	7.99e-5	1.89e-4	7.07e-6
2	2000	0.045	0.064	0.028	4.87e-5	9.52e-5	2.95e-5
2	3000	0.069	0.151	0.036	7.49e-5	2.19e-4	1.72e-5
2	4000	0.068	0.112	0.034	4.52e-5	8.51e-5	7.63e-6
2	5000	0.104	0.233	0.019	6.29e-5	1.85e-4	7.26e-6
3	1000	0.091	0.144	0.037	7.24e-4	1.98e-3	1.58e-4
3	2000	0.066	0.114	0.027	4.52e-4	1.09e-3	7.77e-5
3	3000	0.134	0.167	0.086	3.27e-4	4.83e-4	1.59e-4
3	4000	0.193	0.275	0.123	3.47e-4	4.84e-4	2.78e-4
3	5000	0.152	0.225	0.016	3.28e-4	6.89e-4	4.05e-5
4	1000	0.093	0.141	0.045	5.92e-4	1.14e-3	3.46e-4
4	2000	0.077	0.116	0.040	1.68e-3	3.87e-3	4.20e-4
4	3000	0.150	0.251	0.089	5.64e-4	1.42e-3	2.43e-4
4	4000	0.147	0.272	0.086	8.15e-4	1.55e-3	1.61e-4
4	5000	0.174	0.260	0.136	7.30e-4	8.53e-4	5.35e-4
5	1000	0.111	0.145	0.053	1.71e-3	3.39e-3	3.79e-4
5	2000	0.101	0.134	0.077	2.04e-3	2.72e-3	1.20e-3
5	3000	0.149	0.204	0.098	2.69e-3	3.33e-3	1.22e-3
5	4000	0.092	0.136	0.042	8.84e-4	1.52e-3	3.58e-4
5	5000	0.128	0.170	0.102	1.14e-3	2.80e-3	4.86e-4
6	1000	0.105	0.149	0.079	4.06e-3	6.96e-3	1.27e-3
6	2000	0.104	0.127	0.087	5.19e-3	8.90e-3	2.20e-3
6	3000	0.118	0.155	0.096	4.50e-3	6.04e-3	2.67e-3
6	4000	0.133	0.207	0.097	2.22e-3	4.67e-3	7.85e-4
6	5000	0.146	0.176	0.123	2.23e-3	5.18e-3	1.03e-3
7	1000	0.157	0.188	0.109	9.93e-3	1.33e-2	6.72e-3

larger than six, it was impossible to perform the experiments with the Voronoi diagram due to the lack of the memory space for the Voronoi vertices, as mentioned.

The effect of the search space renormalization scheme is shown in Table II. For $D = 5$ and $N = 5000$, the EA without the scheme was run five times with the same conditions for the two-stage EA (i.e., the same random seeds and the same generations) and was compared to the case with the scheme. We see that the quality of the solution with the two-stage search is better than that of the single stage search; the errors in the costs and the positions of the solutions are reduced by the search space renormalization scheme.

V. CONCLUSION

We have proposed the stochastic search approach using the EA for solving the LES problem on the multidimensional space. The experiments demonstrated the performance of the proposed method in comparison

TABLE II

EFFECT OF THE SEARCH SPACE RENORMALIZATION SCHEME. PERFORMANCE WITH AND WITHOUT THE SCHEME IS COMPARED FOR D=5 AND N=5000.

	E_c (%)			E_p		
	Avg	Max	Min	Avg	Max	Min
Without	0.674	0.921	0.411	1.45e-2	2.20e-2	5.91e-3
With	0.128	0.170	0.102	1.14e-3	2.80e-3	4.86e-4

with the Voronoi diagram approach. The EA approach successfully found the solutions with sufficiently high accuracy. Especially, for the problems in the high-dimensional space, the EA reduces much time in comparison with the Voronoi diagram method. Besides, the EA requires much smaller memory than the Voronoi diagram approach in which the necessary memory space exponentially grows with the dimension.

In addition to the efficiency in terms of the execution time and the memory in high-dimensional spaces, the following advantages are acquired by the use of the proposed stochastic search approach. First, the EA can be easily applied for the problems defined on other spaces such as the surface of the hypersphere. Although some methods constructing the Voronoi diagram on a three-dimensional sphere have been proposed [13], [14], their generalization to the multidimensional hypersphere is not straightforward and no algorithm for this appears to be known. Second, we can easily apply the proposed EA approach to the problem with other distance metrics in the same way given in this paper. Although we have not treated such cases, it is clear that this is possible by just modifying the fitness function in Eq. (2) appropriately. Third, the Voronoi approach inherently limits the location of the center of the empty sphere within the convex hull of the given points, which makes it difficult to solve the problem when the solution is allowed to be outside the convex hull. In the EA, this situation can be treated during the mutation steps by confining the offsprings within the desired domain.

We are currently working on applying the proposed approach to the structural optimization problem of artificial neural networks.

REFERENCES

[1] A. Okabe and A. Suzuki, "Locational optimization problems solved through Voronoi diagrams," *European Journal of Operational Research*, vol. 98, pp. 445-456, 1997.

[2] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, New York: Springer-Verlag, 1985.

[3] M. I. Shamos and D. Hoey, "Closest point problem," in *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, Berkeley, CA, Oct. 1975, pp. 151-162.

[4] G. Voronoi, "Nouvelles applications des paratrés continus à la théorie des formes quadratiques. Deuxième memoie: Recherches sur les parallelloèdres primitifs," *Journal für die reine und Angewandte Mathematik*, vol. 134, pp. 198-287, 1908.

[5] S. J. Fortune, "A sweepline algorithm for Voronoi diagrams," *Algorithmica*, vol. 2, pp. 153-174, 1987.

[6] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine*, New York: IEEE Press, 1995.

[7] X. Yao, *Evolutionary Computation: Theory and Applications*, Singapore: World Scientific Publishing, 1999.

[8] L. J. Park, C. H. Park, C. Park, and T. Lee, "Application of genetic algorithms to parameter estimation of bioprocesses," *Medical and Biological Engineering and Computing*, vol. 35, no. 1, pp. 47-49, Jan. 1997.

[9] J. N. D. Gupta, K. Henning, and F. Werner, "Local search heuristics for two-stage flow shop problems with secondary criterion," *Computers and Operations Research*, vol. 29, no. 2, pp. 123-149, Feb. 2002.

[10] B. Chazelle, "An optimal convex hull algorithm and new results on cuttings," in *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, San Juang, Puerto Rico, Sept. 1991, pp. 29-38.

[11] V. Klee, "On the complexity of d-dimensional Voronoi diagrams," *Archiv der Mathematik*, vol. 34, pp. 75-80, 1980.

[12] C. B. Barber, D. P. Dobkin, and H. T. Huhdanpaa, "The Quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software*, vol. 22, no. 4, pp. 469-483, Dec. 1996. Available: http://www.qhull.org/

[13] K. Q. Brown, "Geometric transforms for fast geometric algorithms," Ph.D. dissertation, Carnegie-Mellon Univ., 1980.

[14] J. M. Augenbaum and C. S. Peskin, "On the construction of the Voronoi mesh on the sphere," *Journal of Computational Physics*, vol. 59, pp. 177-192, 1985.