Journal of Interdisciplinary Mathematics
ISSN: 0972-0502 (Print), ISSN: 2169-012X (Online)
Vol. 27 (2024), No. 7, pp. 1615–1636
DOI : 10.47974/JIM-2002

Touring a sequence of spheres in $\mathcal{R}^3$

Chang-Chien Chou
Department of Information Management
Lunghwa University of Science and Technology
Wanshou Rd., Guishan District
Taoyuan City 33360
Taiwan, R.O.C.

Abstract

How do you touring a sequence of different size of balls and returning to the starting point with a minimum *Euclidean* travelling path? We present an efficient algorithm to answer the question in this paper. One may say that the above mentioned minimum path, a shortest n–gon of n–spheres in three dimensions. Accompanied by three other competing methods: the genetic algorithm, the nearest neighbor algorithm and the random test, all have shown up our proposed algorithm dominates the results in every aspects of spheres distribution. Empirical results have also shown the effectiveness and the quick convergent property about the proposed algorithm.

The proposed algorithm can be conducted for the coverage problem of sensor deployment. In addition, it is also called the shortest n–gon of n–spheres (particles with necessary volume). Since, practitioners in the fields of computational geometry, material science, solid modeling, 3D printing, computer graphics, or any other kinds of CAD/CAM applications may find merits from this paper.

Subject Classification: *14Qxx: Computational aspects in algebraic geometry.*

Keywords: *Sensor deployment, 3D printing, Computer-aided design and manufacturing, Computational geometry.*

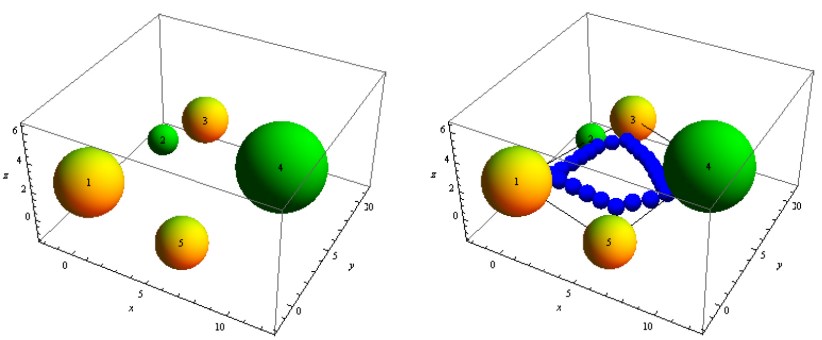
1. Introduction

Cloud and Edge Computing [1] are nowadays everywhere in our daily life. Sensor networks are basic physical architecture in both Cloud and Edge Computing. Intrusion detection is to detect potential malicious sabotage activities on wireless sensor networks which becomes one of

E-mail: samuelchou7@yahoo.com.tw

most popular research topic in sensor networks [23]. In addition to intrusion detection, coverage problems are another crucial issue in wireless sensor networks. Many modern researches devoted themselves in coverage detecting and restoring [2, 26, 27, 31]. The coverage of the wireless base station is usually modeled by sphere in $\mathcal{R}^3$ [5, 30]. Assume there are n standalone stations (disjoint wireless coverage spheres) scattered in space as shown in Fig. 1(a). You are dispatching a fleet of relays connecting them as depicted in Fig. 1(b). For the purpose of providing a reliable fault tolerable connection (i.e., if any or all of the n base stations fail, the communication link will still work. A ring topology is requested.), at least one contact point in the coverage of the base station needs to be contained in at least one relay station. Dispatching a fleet of relays with a minimum number is your mission. The problem in turns to be represented by a geometrical challenge of calculating the *Euclidean* shortest path touring n spherical balls in $\mathcal{R}^3$. This is one version of the *Euclidean* Travelling Salesman Problem (ETSP) in $\mathcal{R}^3$. In order to simplify the problem, the touring sequence is given in this paper. Even doing so, the studied problem is still hard since the points on the surface of the spheres are continuous.

Given a plural number of disjoint spheres (S_i, $i \in \{1,...,n\}$) scattered in a space of $\mathcal{R}^3$. The touring sequence is depicted by i. The path starts from point P_1 on the border of S_1, moves to point P_2 on the border of S_2, ..., S_n, and eventually gets back to the starting P_1on S_1. Among all of the possible points on each sphere forming the paths, the one with the minimum *Euclidean* length (L_{min}) is the goal of the studied problem. The objective function of the studied problem is thus



(a) Stand alone stations

(b) Dispatching a fleet of relays connecting them

Figure 1

Relay placement for connecting stand alone stations

$$L_{min} = \min\ (\sum_{i=1}^{n-1}(\,|P_iP_{i+1}|\ \ _{\forall i \in \{1..n-1\}}\,|) + \ |P_iP_1|\ \ _{\forall\ i=n})\ \forall P_i \in S_i$$

Sphere disjoint implies that the spheres do not overlap, contain, or be tangent to each other. The path is allowed to penetrate the interior of the sphere in wireless sensor networks.

2. Related Works

Many researches discuss the distance of objects in $\mathcal{R}^2$ or $\mathcal{R}^3$. Schwartz [22] proposed means to calculating the distance between two polygons in $\mathcal{R}^3$. Aragón and Molinari [3] turned to the detection phase of contact mechanics. Wolff and Bucher [29] developed an approach to collision detection based on distance fields in $\mathcal{R}^2$ or $\mathcal{R}^3$. Neff [21] addressed a method calculating the distance between two circles in $\mathcal{R}^3$. Kim [17] gave a method calculating the distance of two ellipses in $\mathcal{R}^3$. Lee [18] discussed the method of calculating distance between two curved surfaces in $\mathcal{R}^3$. Ma [20] presented a new method to calculate the shortest path between two canal surfaces by taking a set of cone-spheres as bounding volumes. Ma's method is more effective against Lee's. Efrat [13] and Dror [12] gave a constant-factor approximation algorithm for the ETSP with neighborhood on an arbitrary set of disjoint, connected neighborhoods (circle is one instance of neighborhood) in $\mathcal{R}^2$. However, they are approximation algorithms and the results in $\mathcal{R}^2$ cannot be conducted to $\mathcal{R}^3$ directly. So far, no literature has proposed algorithms along with the experiments for calculating the shortest path connecting n ordered spheres in $\mathcal{R}^3$ as we do.

The author's previous works provides some stuff and foundation for the studied problem. Chou [9] gave a general closed–form root equation to solve the shortest path of three circles in $\mathcal{R}^2$. There are eight general roots in that equation. Chou [8] further reduced the number of roots to two. Since the two roots equation uses the axes rotation accompanying several geometric analyses that is not easy to programming. This paper uses the eight roots equation [9] instead. Chou [6] proposed the algorithm calculating the shortest tour connecting n sequenced circles in $\mathcal{R}^2$. However, the $\mathcal{R}^2$ version cannot be generalized to $\mathcal{R}^3$ directly. Chou [7] further relaxed the constraint "ordered circles" and thoroughly solved the problem calculating the shortest tour of n (non–sequenced) circles in $\mathcal{R}^2$. The algorithm in [7] is constructed based on the general closed–form root equation [9], the local optimization, and the technique of branch and bound with a dominant bound.

3. Algorithms

This section describes the proposed algorithms. Before we can develop our algorithms, some fundamental techniques are introduced in section 3.1. The author's previous works are consolidated by several theorems that can be used as the important components in our proposed algorithms depicted in section 3.2 and 3.3.

3.1 *Fundamental Theorems*

The proposed algorithm requires a local optimization procedure among three consecutive spheres that is the most basic example of our problem where $n = 3$. We now summarize the required stuffs herein by the following theorems.

Theorem 1: *Computing the Euclidean shortest path among three spheres in $\mathcal{R}^3$ can be degenerated to the problem of computing the Euclidean shortest path among three coplanar discs in $\mathcal{R}^2$.*

***Proof*:** Since there is one and only one plane passing through the three centers of the three spheres, any point on the surface of the sphere excluded the boundary of the coplanar great circle can be projected into the interior of the great circle. See Fig. 2 where $\overline{A^*A^{**}}$ is perpendicular to the great circle C_A of sphere S_A. In this fashion, similarly, $\overline{P^*P^{**}} \perp C_C$ and $\overline{B^*B^{**}} \perp C_B$. Therefore, for any point on the face of sphere, such as A^{**}, P^{**} and C^{**}, there exists at least one point on the boundary of C_A, C_C and C_B, such as A, P and B respectively, such that $|\overline{A^*P^*}| < |\overline{A^{**}P^{**}}|$ and $|\overline{B^*P^*}| < |\overline{B^{**}P^{**}}|$. We have

$$|\overline{AP}| + |\overline{BP}| < |\overline{A^*P^*}| + |\overline{B^*P^*}| < |\overline{A^{**}P^{**}}| + |\overline{B^{**}P^{**}}|.$$

To sum up, for constructing a path through any point on the three spheres outside the boundaries of the three great circles, we can find at least one path that is located on the boundaries of the three great circles, thus obtaining a shorter path.

Theorem 2: *Computing the Euclidean shortest path among three circles in $\mathcal{R}^2$ can be reduced to computing the Euclidean shortest path connecting one circle and two points in $\mathcal{R}^2$.*

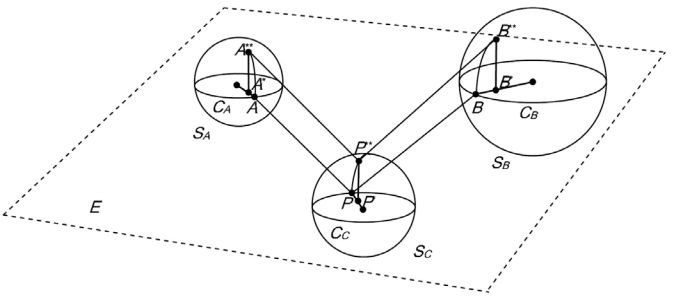


Figure 2

Three spheres, co-plane, and the grate circles

***Proof*:** See [9], omitted. Figure 3 depicts the example of the reduced problem of the shortest path connecting one circle and two points. Computing the shortest path connecting three circles C_1, C_2 and C_3 can be reduced to computing the shortest path connecting one point O_1 to one circle C_2 and then to the other point O_3, i.e., the point–circle–point problem.

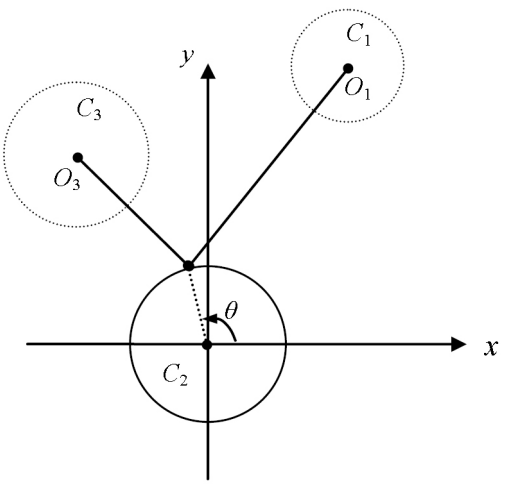


Figure 3

The reduced problem of point–circle–point path calculation

Theorem 3: *Computing the Euclidean shortest path of point-circle-point problem in $\mathcal{R}^2$ can be obtained by a closed–form general roots equation.*

***Proof*:** The closed–form general solution can be found in [9], the proof is omitted. By using this closed–form general solution, simply inputting the coordinates of the two points, the center of the circle as well as its radius, the optimum contact on the boundary of the circle together with its shortest path can be obtained immediately, i.e., the time complexity is O(1).

3.2 *Local Optimization*

After the above theorems were established, the local optimization procedure of three consecutive spheres in $\mathcal{R}^3$ can be obtained as follows. First, the given three consecutive spheres (S_A, S_B, S_C) in $\mathcal{R}^3$ are simplified to a equivalent problem of three consecutive circles in $\mathcal{R}^2$ by using Theorem 1. Second, the reduced problem of computing the shortest path of three consecutive circles in $\mathcal{R}^2$ can be further simplified to computing the shortest path of a point–circle–point problem in $\mathcal{R}^2$ by using Theorem 2. Third, the reduced point–circle–point shortest path calculation can be obtained by applying the closed–form general eight roots equation in [9] directly. The local optimum point for the center circle (the reduced middle sphere, S_B) can be obtained immediately in constant time since it is using a closed–form root equation. Finally, the obtained three optimal points can be restoring back to their initial coordinates system in $\mathcal{R}^3$ by some simple matrix transition calculations. The details are depicted by Algorithm 1 SP3S (shortest path 3 spheres) and Fig. 4. By inputting three consecutive spheres S_A, S_B, and S_C, the results of the shortest path ($path_{min}$) together with its length (L_{min}) can be obtained in constant time O(1).

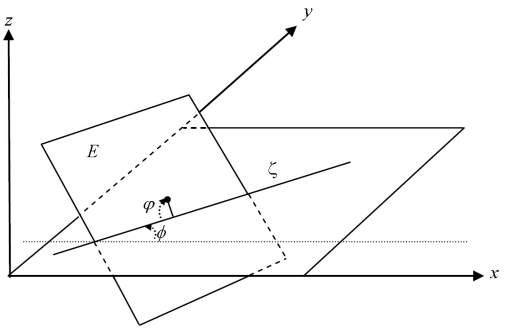


Figure 4

Included angles ϕ, φ and the co–plane E

Algorithm 1. SP3S(S_A, S_B, S_C, $path_{min}$, L_{min})

Input: Three consecutive spheres S_A, S_B, and S_C;

output: Set $path_{min}$ and its length L_{min};

1. Shift the three balls according to the center of S_B coincides with the origin;
2. Compute the co–plane (E) equation passing through S_A, S_B, and S_C;
3. Calculate the intersection equation of xy–plane and E (line ζ);
4. Calculate ϕ, the included angle between x–axis and ζ ;
5. Calculate φ, the included angle between xy–plane and E ;
6. CCW rotate z–axis for ϕ degree and then CCW rotate x'–axis for ϕ degree;
7. Applying the closed–form general roots equation in [9] to obtain the optimum point P'' on the boundary of S_B;
8. CW rotate x''–axis for φ degree and then CW rotate z'–axis for ϕ degree to obtain the results (including P'') to its initial xyz–system;
9. Shift back all results (including P) to their original position;
10. Add the rest two contact points (point A on S_A and point C on S_C) and P to the set $path_{min}$;
11. $L_{min} \leftarrow |path_{min}|$;
12. Return $path_{min}$ and L_{min};

3.3 *Proposed Algorithms*

Herein an efficient local search heuristic is illustrated by Algorithm 2 Steeping_Descent. The idea of Algorithm 2 is straightforward by shortening one line segment at a time. Since step 6 will compute the local optimum point forming the shortest path among three consecutive shperes

$$S_{(j\%n)-1},\ S_{(j\%n)} \text{ and } S_{(j\%n)+1},$$

steps 5 through 6 will yield a shorter segment from $S_{(j\%n)-1}$ to $S_{(j\%n)}$ and then $S_{(j\%n)+1}$. For example, a shorter path goes from S_1 to S_2 and then S_3 if $j = 2$ through steps 5–6.

Algorithm 2. Steeping_Descent (S_i, $path_{\min}$, $L_{\min}$)

Input : n consecutive spheres S_i, $i \in \{1..n\}$. A computational threshold ε. Say, ε=±0.000000000000001;

Output : The shortest path $path_{min}$ with its length L_{min};

1. Randomly draw arbitrary points on the surface of every S_i ;
2. Successively connects those points yielding an initial path with its length l;
3. $j \leftarrow 0$;
4. Do // Looping in a round-robin fashion.
   {
5. Fix contact points on $S_{(j\%n)-1}$ and $S_{(j\%n)+1}$; //local optimize one point.
6. Call SP3S( ) to local optimize the contact of S_j ;
7. Compute path(j) with its length l_j;
8. j++;
9. } until l_{j-1}- l_j <= ε;
10. $path_{min} \leftarrow$ path(j);
11. $L_{min} \leftarrow l_j$;
12. Return $path_{min}$ and L_{min};

However, Algorithm 2 Steeping_Descent adjusts only one contact point in Step 5 in each iteration. We now propose a more robust algorithm that will adjust $n/2$ contact points in each iteration which essentially yields faster convergent speed than that of Algorithm 2.

The principal of the main proposed algorithm comes from the idea of pairwisely local optimization. First, randomly draw one arbitrary point on the boundary of each sphere. Successively connect those points according to the sphere number yielding an initial path. Second, fix all contact points on every even number spheres to local optimize all contact points on each odd number sphere. A shorter path can be obtained. Third, similarly, fix all contact points on every odd number spheres to local optimize all contact points on each even number sphere. Again, another shorter path can be obtained. In this fashion, the loop goes back and forth to local optimize the odd and then the even number of contact points iteratively until a computational threshold is reached. Since the outcome from every odd (or even) local optimization loop constructs a shorter path, the overall

process goes like a squeezing optimization and is hopefully be converged in the shortest path eventually.

The above idea (looping in an odd-even squeezing fashion) is now taken into account in the proposed algorithm illustrated by Algorithm 3 OE3D (odd-even 3 dimension). The input are n consecutive spheres S_i, $i \in \{1...n\}$ with a computational threshold ε ($\varepsilon = \pm 000.0000000000001$ in all of the experiments which follows the IEEE 754 single precision specification). The output is the set of the shortest path $path_{min}$ with its length L_{min}. Where L_j^o denotes the generated tour path length by fixing the even number contact points and then local optimize the odd number contact points in iteration j. On the other hand, L_j^e is to local optimizing the even number contact points in iteration j.

Algorithm 3. OE3D(S_i, $path_{min}$, L_{min})

Input: n consecutive spheres S_i, $i \in \{1..n\}$. A computational threshold $\varepsilon = \pm 000.0000000000001$;

Output: The touring path $path_{min}$ with its length L_{min};

1. Randomly draw one arbitrary point on the surface of each S_i ;
2. Successively connect those points yielding an initial path with its length l_0;
3. $j \leftarrow 1$;
4. Do // Looping in an odd-even squeezing fashion.
   {
5. Fix each even circle's contact point and call SP3S( ) to adjust the contact point on each odd circle yielding a new *path* and its length L_j^o.
6. Fix each odd circle's contact point and call SP3S( ) to adjust the contact point on each even circle yielding another new *path* and its length L_j^e.
7. j++;
8. } until $L_j^o - L_j^e \leq \varepsilon$;
9. $path_{min} \leftarrow path$;
10. $L_{min} \leftarrow L_j^e$;
11. Return $path_{min}$ and L_{min};

The proposed Algorithm OE3D is rapidly shrunk down the tour path in each iteration during steps 4–8. That is, in each iteration, about $n/2$ contact points will be locally optimized. This improves the calculation time against Algorithm 2 Steeping_Descent which adjusts only one contact point in each iteration. By this fashion, the local optimality will come out after the end of each loop. As the yielding path, in each iteration, is shorter than the previous one, this local optimality loop can eventually converge toward the shortest length.

4. Experimental Result

The simulation samples are categorized into two types, Type I has three cases (n=10, n=12 and n=20) with proper spacial volume (10x10x10 or 10x10x12) as shown in the appendices. Type II has two cases (n=30 and n=50) with different spacial volume that is much close to the spacial wireless sensor networks displacement scenario. The scale of altitude is much shorter than that of the *xy*-plan's latitude and longitude. The details of the sample data can be found in the appendices.

4.1. *Competitive Methods*

Since so far there is no other algorithm can be put together for evaluation, we use random selection (RD), the nearest neighbor search (NNS) heuristic, and a general genetic algorithm (GA) for competition.

The random algorithm is simply choosing arbitrary point on the face of each sphere forming the path. As the iteration increases, the chance finding a near optimal path increases accordingly. Assume that computing the sum of each line segment yielding one path length requiring one iteration. The iteration for each random test and the genetic algorithm is 100,000 times, i.e., one hundred thousand paths will be evaluated in the random test as well as in GA search. Under such convention, the NNS requires only one time, i.e., one path only.

The NNS is straightforward. It starts from the first ball. Seek the shortest path connecting to the second ball. And then it searches the shortest path to the 3^{rd} , the 4^{th} , ..., and the n^{th} ball successively. Eventually, the path goes back to the starting point on the first ball completing one tour. Since the properties of fast computing and easy programming, NNS is often been taken into account as a general competitive algorithm in many experiments as well as we do in this paper.

The proposed GA is formulated 100 points (polarpt[,]) on the surface of each ball by the following procedure:

```
for ( i=1, i≤10, i++ )
for ( j=1, j≤10, j++ ) {
   polarpt[ cnt, 1] = r Cos[0.1 * i * 2π ] Cos[0.1 * j *2π ]; //x coordinates
   polarpt[ cnt, 2] = r Sin[0.1 * i * 2π ] Cos[0.1 * j * 2π ]; //y coordinates
   polarpt[ cnt, 3] = r Sin[0.1 * j * 2π ];                    //z coordinates
   cnt++;
};
```

By doing so, the problem is transformed into a bounded combinatorial problem however the 100 numbered points scattered in the surface of each sphere are inadequate, but for the design of gene there should be discrete points on each sphere. The initial chromosome (chro) is chosen by random, for example, n=10 as shown by Table 1.

Table 1

Illustration of chromosome in the gene pool for the case of n = 10.

Spheres	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}
chro 1	88	13	75	48	72	2	99	26	71	65
…	…	…	…	…	…	…	…	…	…	…
chro 10	12	88	50	94	42	75	10	23	10	42

The GA generation is set to be no larger than 100,000 iterations. The fitness function is set to be the objective function, i.e., the length of the tour. We use single point mutation method and the crossover rate is set to be 0.5. The tournament policy is taken to replace weaker genes by the stronger genes in the pool under a probability. The mutation rate is set to be less than 0.1.

4.2 *Empirical Results*

Figure 5 is the result from the proposed OE3D algorithm for the case of n=10 where the red line segments construct the shortest path. Tables 2 and 3 show the results after competing against the random selection, the GA search, and the NNS heuristic algorithm. The OE3D is abbreviated by OE. The random selection algorithm is denoted by RD. The genetic algorithm is depicted by GA. NN represents NNS in Tables 1 and 2. The column "Init. (OE)" represents the initial length of OE. "Worst (RD)" depicts the worst case length of RD. The detailed coordinates of each test are shown in the appendices.

From the empirical results as shown in Table 2, 3, and Fig. 6, we noticed that NNS has better performance on cost (length) and speed (iteration) against random selection as well as GA. This is due to the touring sequence is fixed. Once this constraint is relaxed, random selection and GA may have better performance. Although the speed of OE/NN seems to affirm NNS, the shortest path cannot be obtained by NN is clear. Similarly, while the constraint of "sequenced balls" is relaxed or the number of n becomes larger in follow up researches, NN becomes much more inadequate.

The performance of GA and RD are equally awkward. As the value of n increases, the length of the gene increases accordingly. Different genetic design and policy are required for different gene length. For example, n=10 and n=100 should have different mutation rate, different mutation number of point, different reproduction policy, different gene design, and so on. Therefore, when the length of gene is getting larger, GA design and search becomes more complex and inefficient as well.

4.3 *Analysis*

It is natural that we think as the number of n increases, the calculating time increases accordingly. But one interesting thing that exceeds out of our expectations: the convergent speed in Type II when n=50 (iteration=39) is less than n=30 (iteration=73). We guess it is because the altitude of z–axis of the case n=50 is only 2 (highest – lowest = 3 – 1) that is much smaller

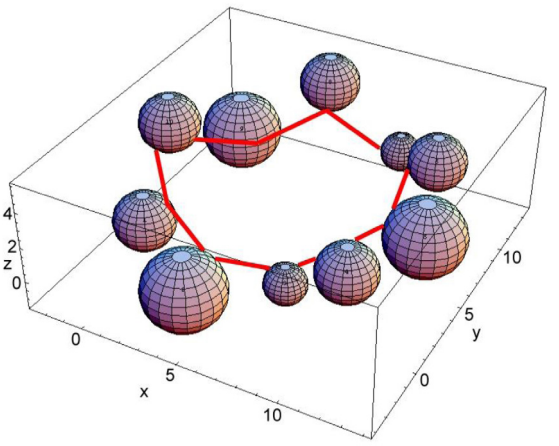


Figure 5

The shortest path connecting 10 spherical balls

than the case of n=30 (highest – lowest = 14 – (–4) = 18). The smaller altitude compresses the algorithm quicker converged in spite of against larger n. However, this may be one of our future works to prove or to explain. In general, if two cases with similar distribution and placement scale, then the case with larger n should consume more iterations to converge like Type I behaved.

The proposed OE3D algorithm clearly outperforms the other three compared methods both on efficiency and effectiveness. The simulation results have affirmed our proposed algorithm on the correctness as well as the performance so that it is capable to be conducted in computer system immediately.

Table 2

Empirical Comparison of Type I

n		Init. (OE) | Worst (RD) length	Final length	Iteration
10	OE	49.84865889962867	36.82745562365802	30
	RD	61.70708230343286	39.26379168336148	100000
	GA	54.25512334088062	39.06105401387353	100000
	NN		40.31149248488908	1
12	OE	125.3897725862249	87.86057838844570	192
	RD	143.0784554873567	99.24380782081482	100000
	GA	126.0612504680716	98.42252773332125	100000
	NN	92.00787142025924		1
20	OE	214.3009852366947	170.8890823654210	244
	RD	246.1235391779199	194.3442783072895	100000
	GA	221.8772281234093	196.1852395534580	100000
	NN	179.6861388489830		1

4.4 *Time Complexity*

Some may say that more number of n is required for simulation. However, under the same volume of space, when the number of sphere is getting larger, sphere is more like point with respect to the same volume of space. That is, small or medium number of spheres with identical spacial volume is better for the studied problem. Since, when n becomes very large, sphere is much more like point, there need not to use any algorithm. Simply connect those points (or spheres) successively is enough. Sphere volume is no longer an issue under such scenario. You are

Table 3

Empirical Comparison of Type II

n		Init. (OE) | Worst (RD) length	Final length	Iteration
30	OE	255.0538777943137	204.1502385065212	73
	RD	277.6480148318444	225.8146506143545	100000
	GA	261.0257548196648	225.7958306812185	100000
	NN	213.0268753289819		1
50	OE	324.7948691417741	171.5232692393156	39
	RD	346.2900976532468	262.2330107773236	100000
	GA	301.6317158712138	272.5521258546835	100000
	NN	189.0549020065347		1

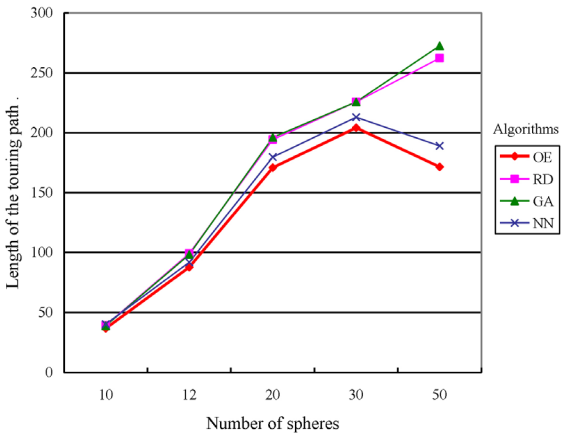


Figure 6

The proposed OE3D algorithm yields the shortest path

doing typical travelling salesman problem when the "ordered sphere" constraint is relaxed.

On the other hand, since the problem is not combinatorial, the stop criterion relies on the requirement of precision only. The more digit of precision you requested, the more iteration the algorithm performs. Therefore, the time complexity has loosely relationships with the number of n.

5. Conclusion

We consider the problem of calculating the *Euclidean* shortest path connecting n spheres in $\mathcal{R}^3$. You may say the problem is to find a shortest n–gon of n–spheres. The wireless sensor networks is one application of the studied problem. Where there are n sequenced standalone wireless stations with their coverages are represented by spheres in $\mathcal{R}^3$. You are planning to dispatching a fleet of relays to connect them as a ring under a restriction of been sent of minimum number of relays. In view of the author's previous works, this paper proposed an efficient algorithm for the studied problem. Empirical results have shown the correctness as well as the performance of the proposed algorithm. Empirical results also indicate the proposed OE3D algorithm is easy to programming. Hence, the algorithm can be conducted to the computer system directly.

5.1 *Applications*

The proposed algorithms can deal with the problem of relay placement as mentioned in Section 1. Meanwhile, practitioners in the fields of rapid prototyping, 3D printing and/or other computer aided design and manufacturing [25, 28] may find merits from this paper.

In addition, there exists a variety of potential applications that can consider conducting the results of this paper: layered manufacturing need to calculate the distance between spherical surfaces [11, 20, 32] and then determine a tool path to machining them; 3D-printing is a new technology with versatile implications [14, 24]; Ray tracing in geophysics [4, 15]; Computer graphics [10]; Molecular geometry has the applications to calculate the distance among spherical molecules [16]. The results of this work can be conducted in such CAD/CAM systems [19] immediately because of its quick convergent ability and the facile programmability.

5.2 *Future Works*

The future works can try to relax the constraint of "ordered spheres". Moreover, once the path is prohibited from piercing through the interior of the ball in some applications, the path needs to make an additional detour on the surface of the ball. This will fluctuate the contact point on each sphere making the algorithm more complicated. And then becomes another interesting topic in the future.

In addition, despite of the empirical results, the proposed OE3D algorithm is suggested to have a theorem with proof to theoretically

consolidate its correctness as well as the time complexity. However, the convergent time complexity is not easy to estimate since it is not a purely combinatorial problem. The stop criterion is correspondent with the computational threshold as well as the required precision about π.

Furthermore, since ellipsoid is the generalization of sphere, once the algorithm on touring a plural number of ellipsoid can be obtained, the applications are then definitely broadened.

Ackwledgement

This work is supported by the National Science and Technology Council, Taiwan, R.O.C. under the grant NSTC 112-2221-E-262-005

Appendices

Appendix A: Coordinates of test data for the case of n=10. After 30 iterations performed by our OE3D algorithm, we come out the shortest path length, 36.82745562365802.

n	x	y	z	r	n	x	y	z	r
1	0	1	1	1.5	6	10	12	1	1.5
2	4	-2	0.5	2	7	8	12	1	1
3	8	0	1	1	8	4	12	4	1.5
4	10	2	1	1.5	9	0	10	1	2
5	12	6	1	2	10	-1	5	4	1.5

Appendix B: Coordinates of test data for the case of n=12. After 192 iterations performed by our OE3D algorithm, we come out the shortest path length, 87.86057838844570.

n	x	y	z	r	n	x	y	z	r
1	2	2	2	2	7	10	2	10	2
2	10	2	2	1.5	8	2	2	10	1.5
3	10	10	2	2.5	9	-5	6	4	1.5
4	2	10	2	2	10	6	14	7	2
5	2	10	10	1.5	11	15	5	4	1.5
6	10	10	10	2.5	12	5	-6	7	2.5

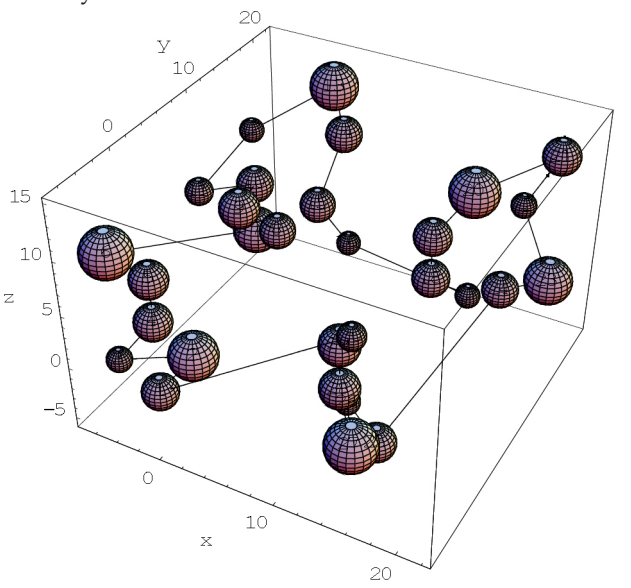
Appendix C. Coordinates of test data for the case of n=20. After 244 iterations performed by our OE3D algorithm, we come out the shortest path length, 170.8890823654210.

n	x	y	z	r	n	x	y	z	*r*
1	2	2	2	2.5	11	15	5	4	1.5
2	10	2	2	2	12	5	-6	7	2
3	10	10	2	1.5	13	14	-6	8	2.5
4	2	10	2	1	14	10	-6	1	2
5	2	10	10	1.5	15	-6	-6	1	1.5
6	10	10	10	2	16	-6	-6	10	1
7	10	2	10	2.5	17	-4	2	10	1.5
8	2	2	10	2	18	-6	14	10	2
9	-5	6	4	1.5	19	14	14	10	2.5
10	6	14	7	1	20	14	14	1	1.5

Appendix D1: Coordinates of test data for the case of n=30. After 73 iterations performed by our OE3D algorithm, we come out the shortest path length, 204.1502385065212.

n	*x*	*y*	*z*	*r*	*n*	*x*	*y*	*z*	*r*
1	-3	-3	-4	1.5	16	3	17	0	1
2	-1	-1	-1	2	17	15	15	0	1
3	-5	-5	0	1	18	13	12	3	1.5
4	-3	-3	3	1.5	19	13	12	7	1.5
5	-3	-3	7	1.5	20	15	15	10	2
6	-5	-5	10	2	21	20	20	12	1.5
7	-5	15	0	2	22	18	18	8	1
8	-1	12	3	1.5	23	20	20	0	2
9	2	2	12	1.5	24	17	17	0	1.5
10	-3	12	7	1.5	25	15	-1	-3	1.5
11	0	0	14	1	26	12	0	-1	1
12	-5	15	10	1	27	15	-5	0	2
13	0	20	13	2	28	13	-3	3	1.5
14	2	18	10	1.5	29	13	-3	7	1.5
15	0	17	3	1.5	30	15	-5	10	1

Appendix D2. Layout of test data for the case of n=30



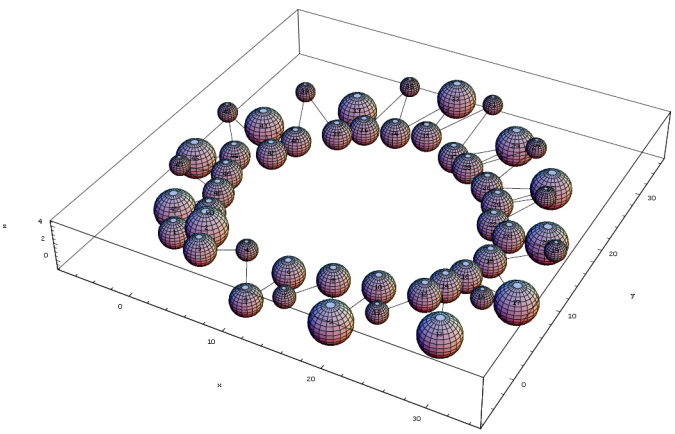
Appendix E1: Coordinates of test data for the case of n=50. After 39 iterations performed by our OE3D algorithm, we come out the shortest path length, 171.5232692393156.

n	x	y	z	r	n	x	y	z	r
1	0	1	1	1.5	26	23	20	2	1.5
2	3	2	2	2	27	25	27	3	1
3	5	-2	3	1.5	28	20	22	2	1.5
4	8	1	2	1	29	22	29	1	2
5	11	-4	1	1.5	30	18	23	2	1.5
6	13	0	2	1.5	31	18	32	3	1
7	15	-4	3	1	32	14	25	2	1.5
8	17	1	2	1.5	33	13	34	1	2
9	19	-3	1	2	34	11	24	2	1.5
10	21	2	2	1.5	35	9	31	3	1
11	23	-2	3	1	36	8	23	2	1.5
12	25	3	2	1.5	37	5	27	1	2
13	28	0	1	2	38	6	21	2	1.5
14	26	5	2	1.5	39	0	25	3	1
15	30	4	3	1	40	3	18	2	1.5

Contd...

16	27	7	2	1.5	41	-2	20	1	2
17	32	7	1	2	42	2	15	2	1.5
18	28	10	2	1.5	43	-5	18	3	1
19	33	13	3	1	44	-1	13	2	1.5
20	28	14	2	1.5	45	-5	12	1	2
21	31	16	1	2	46	0	10	2	1.5
22	26	15	2	1.5	47	-4	8	3	1
23	29	20	3	1	48	1	7	2	1.5
24	25	18	2	1.5	49	-2	4	1	2
25	28	24	1	2	50	2	4	2	1.5

Appendix E2: Layout of test data for the case of n=50



References

[1] K. Agrawal and P. Khetarpal, "Computational intelligence in edge and cloud computing," *J. Inf. Optim. Sci.,* vol. 43, no. 3, pp. 607–613 (2022).

[2] T. Amgoth and P. K. Jana, "Coverage hole detection and restoration algorithm for wireless sensor networks," *Peer-to-Peer Netw. Appl.*, vol. 10, no. 1, pp. 66–78 (2017).

[3] A. M. Aragón and J. F. Molinari, "A hierarchical detection framework for computational contact mechanics," *Comput. Methods Appl. Mech. Eng.,* vol. 268, pp. 574–588 (2014).

[4] C. Bai, S. Greenhalgh, and B. Zhou, "3D ray tracing using a modified shortest-path method," Geophysics, vol. 72, no. 4, pp. T27–T36 (2007).

[5] H. Chen, X. Wang, B. Ge, T. Zhang, and Z. Zhu, "A multi-strategy improved sparrow search algorithm for coverage optimization in a WSN," Sensors, vol. 23, no. 8, p. 4124 (2023).

[6] C. C. Chou, "An efficient algorithm for relay placement in a ring sensor networks," Expert Syst. Appl., vol. 37, no. 7, pp. 4830–4841 (2010).

[7] C. C. Chou, "On the shortest path touring n circles," *Int. J. Adv. Comput. Technol.,* vol. 4, no. 10, pp. 356–364 (2012).

[8] C. C. Chou, Y. K. Chen, and S. Y. Chou, "A fundamental tool path planning problem for circles in layered manufacturing," *Integr. Comput.-Aided Eng.*, vol. 15, no. 1, pp. 37–52 (2008).

[9] S. Y. Chou, C. C. Chou, and Y. K. Chen, "A base function for generating contour traversal paths in stereolithography apparatus applications," Expert Syst. Appl., vol. 35, no. 1–2, pp. 235–244 (2008).

[10] K. C. Ciesielski, R. Strand, F. Malmberg, and P. K. Saha, "Efficient algorithm for finding the exact minimum barrier distance," *Comput. Vis. Image Understand.,* vol. 123, pp. 53–64 (2014).

[11] A. F. Cook IV and C. Wenk, "Shortest path problems on a polyhedral surface," Algorithmica, vol. 69, no. 1, pp. 58–77 (2014).

[12] M. Dror, A. Efrat, A. Lubiw, and J. S. B. Mitchell, "Touring a sequence of polygons," in Proc. 35th ACM Symp. Theory of Comput., San Diego, CA, USA, pp. 473–482 (2003).

[13] A. Efrat, S. P. Fekete, P. R. Gaddehosur, J. S. B. Mitchell, V. Polishchuk, and J. Suomela, "Improved approximation algorithms for relay placement," in Lecture Notes Comput. Sci., vol. 5193, pp. 356–367 (2008).

[14] S. M. Feeman, L. B. Wright, J. L. Salmon, and H. Dodziuk, "Exploration and evaluation of CAD modeling in virtual reality," *Comput.-Aided Design Appl.,* vol. 15, no. 6, pp. 892–904 (2018).

[15] B. Giroux and B. Larouche, "Task-parallel implementation of 3D shortest path raytracing for geophysical applications," *Comput. Geosci.,* vol. 54, pp. 130–141 (2013).

[16] D. Hall, S. Li, K. Yamashita, R. Azuma, J. A. Carver, and D. M. Standley, "A novel protein distance matrix based on the minimum arc-length between two amino-acid residues on the surface of a globular protein," Biophys. Chem., vol. 190–191, pp. 50–55 (2014).

[17] I. S. Kim, "An algorithm for finding the distance between two ellipses," *Commun. Korean Math. Soc.*, vol. 21, no. 3, pp. 559–567 (2006).

[18] K. Lee, J. K. Seong, K. J. Kim, and S. J. Hong, "Minimum distance between two sphere-swept surfaces," Comput.-Aided Des., vol. 39, pp. 452–459 (2007).

[19] L. Liberti, C. Lavor, N. Maculan, and A. Mucherino, "Euclidean distance geometry and applications," SIAM Rev., vol. 56, no. 1, pp. 3–69 (2014).

[20] Y. Ma, C. Tu, and W. Wang, "Distance computation for canal surfaces using cone-sphere bounding volumes," Comput. Aided Geom. Des., vol. 29, no. 5, pp. 255–264 (2012).

[21] C. A. Neff, "Finding the distance between two circles in three-dimensional space," IBM J. Res. Dev., vol. 34, no. 5, pp. 770–775 (1990).

[22] J. T. Schwartz, "Finding the minimum distance between two convex polygons," Inf. Process. Lett., vol. 13, no. 4–5, pp. 168–170 (1981).

[23] N. Singh and D. Virmani, "Computational method to prove efficacy of datasets," J. Inf. Optim. Sci., vol. 42, no. 1, pp. 211–233 (2021).

[24] B. R. Sundar, A. Chunduru, R. Tiwari, A. Gupta, and R. Muthuganapathy, "Footpoint distance as a measure of distance computation between curves and surfaces," Comput. Graph., vol. 38, pp. 300–309 (2014).

[25] M. A. F. Vicente, A. Goncalves, and J. Vitoria, "Euclidean distance between two linear varieties," Appl. Math. Sci., vol. 8, no. 21, pp. 1039–1043 (2014).

[26] X. Wang, T. Nie, and D. Zhu, "Indoor robot path planning assisted by wireless network," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, p. 123 (2019).

[27] Y. Wang, S. Wu, X. Gao, F. Wu, and G. Chen, "Minimizing mobile sensor movements to form a line K-coverage," Peer-to-Peer Netw. Appl., vol. 10, no. 4, pp. 1063–1078 (2017).

[28] X. Wei and A. Joneja, "On computing the shortest path in a multiply-connected domain having curved boundaries," Comput.-Aided Des., vol. 48, pp. 39–41 (2014).

[29] S. Wolff and C. Bucher, "Distance fields on unstructured grids: Stable interpolation, assumed gradients, collision detection and gap function," *Comput. Methods Appl. Mech. Eng.*, vol. 259, pp. 77–92 (2013).

[30] H. Xu, B. Wang, J. Song, H. Hong, and X. Zhang, "An algorithm for calculating coverage rate of WSNs based on geometry decomposition approach," Peer-to-Peer Netw. Appl., vol. 12, pp. 568–576 (2019).

[31] R. Xu, "Path planning of mobile robot based on multi-sensor information fusion," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, p. 44 (2019).

[32] L. Zhu, H. Ding, and Y. Xiong, "Simultaneous optimization of tool path and shape for five-axis flank milling," *Comput.-Aided Des.,* vol. 44, no. 12, pp. 1229–1234 (2012).

Received November, 2023