International Journal of Computational Intelligence Systems
Vol. **14(1)**, 2021, *pp.* 228–237
DOI: https://doi.org/10.2991/ijcis.d.201127.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

Research Article

Parallel DNA Algorithms of Generalized Traveling Salesman Problem-Based Bioinspired Computing Model

Xiaomin Ren[1], Xiaoming Wang[1], Zhaocai Wang[1,*], Tunhua Wu[2,*]

[1]*College of Information, Shanghai Ocean University, Shanghai, 201306, P. R. China*
[2]*School of Information Engineering, Wenzhou Business College, Wenzhou, 325035, P. R. China*

ARTICLE INFO

Article History
Received 21 May 2020
Accepted 18 Nov 2020

Keywords
DNA computing
Adleman–Lipton model
Generalized traveling salesman problem
NP-hard problem

ABSTRACT

Generalized traveling salesman problem (GTSP) is a classical combinatorial optimization problem, in which the optimization goal is the minimum route combination. Since the GTSP is a more complex problem than the traveling salesman problem (TSP), the GTSP can be considered an extension of the TSP. At present, compared with TSP, GTSP has been widely used in practice. In GTSP, n nodes are divided into m clusters, and the problem is divided into two categories according to different constraints. According to the second kind of the GTSP, we propose a parallel biochemical algorithm to solve it. We use deoxyribonucleic acid (DNA) biological strands to represent different vertices, point groups and weights, and get the optimal solutions to a series of different biochemical reaction combinations of DNA sequences. Compared with other algorithms, our algorithm reduces the time complexity to $O(n^2)$, and proves the feasibility.

1. INTRODUCTION

It is well known that the generalized traveling salesman problem (GTSP) is a generalization of the traveling salesman problem (TSP), and therefore is a nondeterministic polynomial (NP) problem. TSP is a commodity salesman from a city to several cities to sell commodities, and he needs to back home after all. The question is how he should arrange the route to minimize the cost of the tour. In the language of graph theory, given a connected graph G (each side has nonnegative weight), a loop is required to visit each node only once and return the initial vertex, and then the total weight is minimized. Different from TSP, the vertices in GTSP are divided into groups and don't need to traverse every vertex. GTSP is generally divided into two categories: the first type requires that only one vertex in each point group be accessed (Figure 1) and the second type requires the loop to access at least one vertex of each point group (Figure 2). Then, the GTSP is defined as finding a special Hamilton cycle in a complete graph that accesses each point group and returns the weight of the initial vertices and the minimum. This paper discusses the second type of GTSP, which requires that point groups disjoint, and each vertex is accessed at most once.

The GTSP can be described as follows: let a complete undirected weighted graph $G = (V, E, W)$, where $V = \{v_1, v_2, \cdots, v_n\}$ is the vertex set, $E = \{e_{ij} | v_i, v_j \in V\}$ is the edge set, $W = \{w_{ij} | w_{ij} \geq 0, w_{ii} = 0, i, j \in V\}$ is the weight set. In practical problems, weights can be interpreted as distance, time, etc. The set of points is divided into

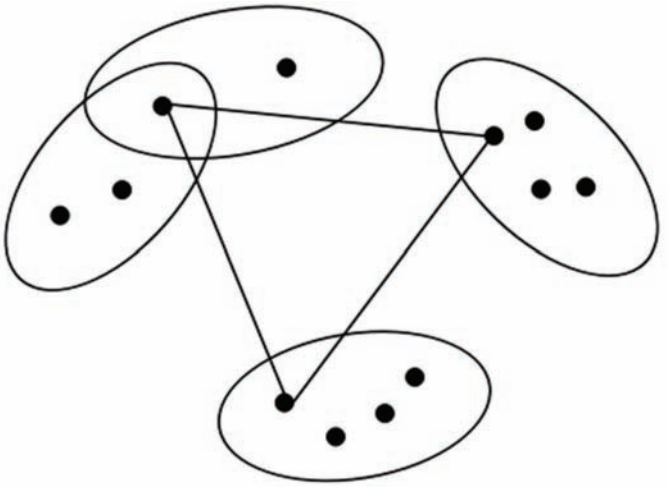

Figure 1 The first kind of generalized traveling salesman problem (GTSP).

m groups $V_1, V_2, \cdots, V_m$, satisfy: $V_i \cap V_j = \varnothing$ for all $i \neq j$ and $V = V_1 \bigcup V_2 \bigcup \cdots \bigcup V_m$.

The object function of the GTSP is as follows:

$$Min\ D = \sum_{i=1}^{m-1} w_{p_i p_i+1} + w_{p_m p_1} \quad v_{p_i} \in V_i$$

In the 1960s, GTSP was initially proposed by Henry-Labordere [1], Saksena [2], and Srivastava *et al.* [3] almost simultaneously. So far,

*Corresponding authors. Email: zcwang1028@163.com; appll188@163.com

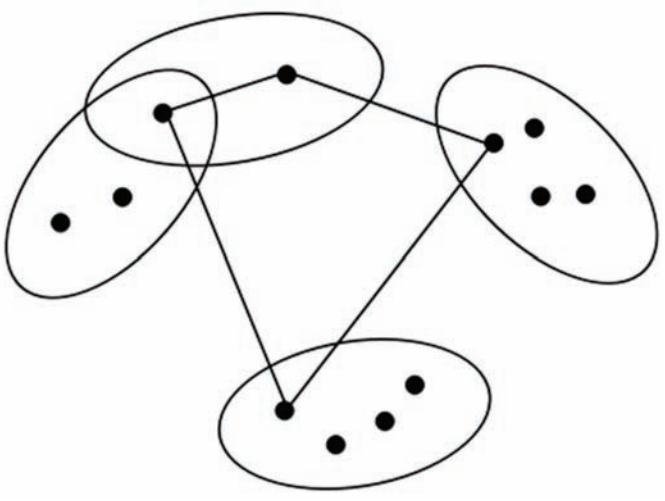

Figure 2 The second kind of generalized traveling salesman problem (GTSP).

there are many practical applications for this problem. Logistics system designs [4], email [5], route planning and naval fleet maritime supply planning [6] are all real-life scenarios of GTSP.

For the first type of GTSP solution, Henry-Labordere, Saksena and Srivastava [1–3] introduced the method of solving GTSP with a simple dynamic programming method. Still, the calculation efficiency of these methods was very low, which can only be used in the case of small problem scale. Then Laporte and Nobert [5] proposed an integer linear programming method with symmetric distance matrix and evaluated the performance of the algorithm using experimental data. This algorithm can solve larger problems than dynamic programming. Laporte *et al.* [7] discussed the asymmetric GTSP, using relaxation constraints to ensure access to all point groups, and then expanding or modifying the asymmetric TSP by branch and bound algorithm. This improved method increased the problem size to 104 vertices by calculating the results. Fischetti *et al.* [8] proposed a solution to accurately solve GTSP by using branch and cut, which provides the best solution for an instance of 442 vertices.

Since there are many sophisticated algorithms for TSP solution, some researchers convert GTSP into TSP and then solve it. The current optimization algorithms used to solve TSP problems, such as differential algorithm [9], particle swarm algorithm [10] and ant colony algorithm [11]. Among them, the difference algorithm is more suitable for solving continuous optimization problems and has strong stability. The particle swarm algorithm improves the local search ability of the algorithm when solving TSP, and can better obtain accurate solutions. The ant colony algorithm has also been improved in terms of convergence speed and solution accuracy. Many scholars focus on how to transform GTSP into TSP. For example, Noon and Beat [12] redefined the arc and arc cost of GTSP according to the rules and added a loop within the zero-cost cluster to convert GTSP into an equivalent TSP. Dimitrijevic and Saric [13] proposed to transform the GTSP into an asymmetric TSP with $2n$ cities, in which the route between the vertices of the same point group is directed, and this algorithm expanded the number of vertices to 2 times. Li and Zhao [14] proposed the method of separating the intersected GTSP and effectively transforming the separated GTSP. However, the results of these algorithms are not ideal, and the efficiency is too low when dealing with large-scale problems. And we need to solve the exact solution from the obtained TSP examples; otherwise, it will not be the optimal solution to GTSP.

An effective algorithm will run for a long time as the number of problem nodes or point groups increases. Later, researchers proposed several genetic algorithms (GA) to solve GTSP, and some of them combined local search algorithms with GA to obtain better-operating efficiency. The generalized chromosome genetic algorithm (GCGA) proposed by Wu *et al.* [15] is by using mixed coding to distinguish the vertices in different groups, and the algorithm generates various feasible solutions through some operations between chromosomes such as crossing and mutation. It then obtains the optimal solution by comparing the loop cost with the function. And in this paper, GCGA is used to solve some examples of processing simple geometric shapes in rectangular plates. In 2006, for the second type of GTSP, Zhao *et al.* [16] proposed a new GA to add virtual vertices to the generalized chromosome, in which the information of the shortest route between each pair of vertices is reserved in an additional matrix. Wang *et al.* [17] improved crossover and mutation operators based on the GCGA algorithm and designed a new coding method to obtain a hybrid chromosome genetic algorithm (HCGA). The HCGA uses binary mixed encoding to change the head crossing and increase the possibility of multiple offspring, so the global search effect is better. Snyder and Daskin [18] proposed an effective heuristic algorithm based on the GA. This algorithm improves the solution through random key encoding, and compared with other algorithms improves the quality and running time of the solution in solving the TSP. Ardalan *et al.* [19] solved the GTSP by making appropriate modifications to the imperialist competition algorithm, using the new coding changes the impact on the cluster and the node selection to produce different results. It has better performance than other algorithms in the benchmark tests of 27 and 40 instances. Kan and Zhang [20] designed an individual mutation method to change the operation of the ant colony optimization algorithm to solve GTSP, and reduced the execution frequency and time complexity of the routing algorithm. Smith and Imeson [21] conducted a research based on adaptive large neighborhood search heuristics. This algorithm has been implemented, and we can find a better solution based on various existing and new problems compared with well-known algorithm tests. Mehdi *et al.* [22] improved the breakthrough local search (BLS) meta-heuristic and GA to reduce the running time. Experiments show that the improved algorithm can successfully obtain the optimal solution from most research examples.

Up to now, there are many studies on the first kind of GTSP. There are few types of research on the second kind of GTSP. And with the expansion of the scale of the problem, it will become more difficult to solve the problem in polynomial time. On the other hand, DNA computing, as a new intelligent computing algorithm, has two important advantages: huge storage capacity and a large amount of parallelism. A large number of parallel operations can cause chemical reactions of small molecules, and billions of operations can be carried out simultaneously. As a carrier of information, DNA molecules have huge computational storage capacity. Compared with the exponential complexity of other classical algorithms, DNA computing can solve complex optimization problems in polynomial time complexity. Therefore, using DNA computing to solve GSTP will be a meaningful attempt.

We need to obtain an optimal route from the set of all possible routes, and this optimal solution usually has the following restrictions:

1. All routes are continuous, starting from the specified vertex and ending;
2. At least one vertex is accessed in each point group, and each vertex is accessed at most once;
3. The sum of the weights of all routes is the smallest.

Figure 3 shows an example of a graph theory model for a GTSP, in which the eight vertices are divided into three different point groups. In Figure 3, we assume that the travel agent starts from the v_1 node. Obviously, after logical calculation, the shortest routes that meet the requirements are: $v_1 \rightarrow v_7 \rightarrow v_6 \rightarrow v_1$ and $v_1 \rightarrow v_6 \rightarrow v_7 \rightarrow v_1$. The total length of the route is 5.

In this paper, for the second kind of GTSP, we propose a DNA algorithm based on the Adleman–Lipton model to solve the GTSP with $O(n^2)$ time computing complexity. Compared with the previous algorithms, the computational complexity of the algorithm is greatly reduced, and the feasibility can also be verified by theory and simulation experiments.

The rest of the paper is organized as follows: In Section 2, we review the development of DNA computing and introduce the specific operations of the Adleman–Lipton model. The algorithms and detailed operations of GTSP are presented in Section 3. In Section 4, the feasibility and time complexity of the algorithms are analyzed. Section 5 gives the simulation results of DNA algorithms. The advantages of the DNA algorithm research prospects are considered in Section 6.

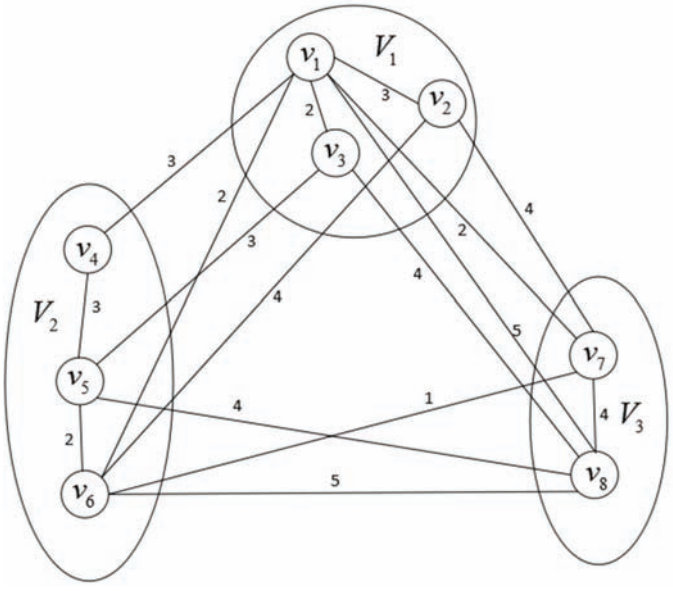

Figure 3 An example of the generalized traveling salesman problem (GTSP) with 8 vertices and 3 groups.

2. BACKGROUND KNOWLEDGE

2.1. DNA Computing

DNA molecules in living cells are the leading stores of information. Countless years of biological evolution have improved the molecules and enzymes in the body that can both copy the information in DNA and pass it on to other molecules. DNA is a double-stranded polymer composed of tandem deoxynucleotide chains, where deoxynucleotides are composed of deoxyribose, phosphoric acid and nitrogen-containing bases. The diversity of DNA is due to the random arrangement of base pairs. Different nucleotides base by its definition, respectively for *A* (adenine), *G* (guanine), *C* (cytosine) and *T* (thymine). The four types of bases have Watson–Crick complementarity, that is, two relative bases conform to complementary characteristics, in which *T* matches *A* and *C* matches *G*. Base pairing is the basis of DNA double helix structure and the basis of replication, transcription and translation. Under the right conditions, two single-stranded DNA can form a double strand, and the number of nucleotides in a single strand is the length of the single-stranded DNA.

DNA computing emerged with the rise and development of molecular biology. Biological computing is done by taking advantage of the structure and function of DNA molecules and the parallel operations in their interactions. In the process of continuous development, the advantages of storage capacity and energy consumption gradually emerge. Massive parallelism occurs because molecules interact chemically in tiny volumes and can perform billions of operations simultaneously. Because of the DNA molecule as a carrier of information, calculation of storage capacity is huge. At the same time, the enzymes that perform DNA calculations have been produced over thousands of years of biological evolution and are highly energy efficient. DNA has a broad prospect of resource application. In 1994, Adleman [23] carried out a pioneering work using DNA computing to solve a well-known computational difficulty, namely the directed Hamiltonian cycle problem. As the number of variables increases, the problem becomes more and more difficult. But using DNA computing, the problem can be easy to be solved. The finding proves the feasibility of DNA computing to solve specific problems. Its novelty lies in setting a precedent for applying mathematical problems to calculations at the molecular level, and his pioneering work inspires people. After that, Lipton [24] solved the satisfiability problem based on Adleman's experiment. Qi Ouyang *et al.* [25] solved the maximum mass problem by using molecular biology techniques. Head *et al.* [26] solved the problem of maximum independent subset. So far, scientists devoted themselves to the research field of DNA computing and proposed many DNA computing models for NP problems in combinatorial optimization, such as the strand replacement model [27–29], self-assembly model [30,31], tile model [32,33], nanoparticles model [34–36], etc. GA can be used as a tunnel for DNA computing to transform complex optimization problems. Now, with the development of science and technology, GA has been combined with other evolutionary algorithms, such as the gravitational search algorithms (GSAs), particle swarm optimization (PSO) [37,38], etc. These hybrid algorithms have a better hierarchical search function and higher efficiency. At the same time, DNA computing has become the focus of new computing models, which can provide strong technical support for many problems that fail to have effective solutions.

2.2. The Adleman–Lipton Model

Suppose that a series of test tubes contain a limited number of DNA molecules of composed of $\{A, G, C, T\}$. The following is the operation of the biological experiment.

1. *Merge* (T_1, T_2): the strands of test tubes T_1, T_2 are mixed in tube T_1, and the tube T_2 is empty;
2. *Annealing (T)*: it can produce all feasible double strands in T, and still store them in T after annealing;
3. *Denaturation (T)*: it can dissociate each double strand in tube T into two corresponding single strands;
4. *Separation* (T_1, X, T_2): for a test tube T_1 and a set of strings X, it removes all single strands containing X from tube T_1, and produces a test tube T_2 with the removed strands;
5. *Discard (T)*: for a test tube T, it discards the tube T;
6. *Copy* (T_1, T_2): given a test tube T_1, it can copy all the biological strands from tube T_1 to test tube T_2;
7. *Append (T, Z)*: given a test tube T and short singled DNA strand Z, it appends Z at the end of every strand in the tube T;
8. *Selection* (T_1, L, T_2): given a test tube T_1 and an integer L, it moves all biological strands of L-length from tube T_1 to test tube T_2, and the remaining biological strands are still in tube T_1;
9. *Sort* (T_1, T_2, T_3): given a test tube T_1, it can move the shortest biological strands to test tube T_2, the longest biological strands to test tube T_3, and the remaining strands still in tube T_1;
10. *Read (T)*: given a test tube T, it recognizes composition of biological strands in tube T.

Although DNA computing differs from traditional intelligent methods, their computational thinking is the same. Since the above operations can be completed in a limited experimental step, the complexity of each operation can be reasonably set to $O(1)$ time [39–43].

3. DNA ALGORITHMS FOR THE GTSP

3.1. Preliminary Ideas

General ideas for conducting experiments using DNA operations are as follows: first, the vertices and point groups corresponding to the GTSP are represented by specific symbols; then, all possible DNA chains of the problem are generated by the biochemical reactions, and feasible chains are selected according to whether different constraints are met; finally, the biological strings corresponding to the optimal solution are obtained by searching and identifying.

It is divided into four steps:

Step 1: Construct all possible random routes from the specified fixed node to the end of the same node;

Step 2: For all feasible solutions, the route passing through each point group at least once, in turn, is filtered;

Step 3: Ensure that each route does not repeat through a node, so that it traverses all point groups once and each vertex at most once;

Step 4: By comparing the weights of each route, the optimal result of the GTSP is obtained.

3.2. Notations and Symbols

In this paper, in order to clarify and standardize the expression of our algorithm, it is necessary to define and explain the symbols and notations in the algorithm. Therefore, we use the symbols in Table 1 to illustrate.

3.3. Graph Theory Expression

In the following, we set symbols $\#, A_1, B_1 (i \in \{1, 2, \cdots, n\})$, $p_i (p_i \in \{1, 2, \cdots, m\})$ to represent DNA different strands with the same length, which we make symbols for t mer length. Obviously, the longer the length of single-stranded DNA synthesis show that the more complicated problems. Then, we use a different single-stranded DNA symbol $A_i p_i B_i$ for vertex v_i, where p_i means v_i is contained in the group of p_i points. The symbol # indicates the beginning and ending of the DNA strand. At the same time, using their complementary strand $\overline{B_i A_j}$ $(e_{ij} \in E)$ synthetic double-stranded. Also, to calculate the weights of the different routes through the nodes, we designed the DNA string X and ψ with the length of t mer. Generally, we assume that v_1 is the starting and ending point of the route.

3.4. Detailed DNA Algorithms

We will take the GTSP in Figure 3 as an example to introduce the algorithm process in detail.

3.4.1. Generate various possible chains

For a graph with m point groups, where n vertices are distributed in these point groups. We specify to start from v_1 and traverse different vertices to form all possible routes. We set:

$T_1 = \{\#A_1 p_1 B_1, A_2 p_2 B_2, \cdots, A_i p_i B_i, \cdots, A_n p_n B_n, A_1 p_1 B_1 \# | v_i \in V, p_i \in \{1, 2, \cdots, m\}\}$;

$T_2 = \{\overline{B_i A_j} | e_{ij} \in E\}$;

$T_3 = \{X\}$.

Table 1 Notations and symbols.

Symbol	Description	Symbol	Description
V	Vertex set	E	Edge set
W	Weight set	v_i	The i-th vertex
V_i	The i-th point group	n	Number of vertices
m	Number of point groups	#	End of DNA strand
A_i, B_i	DNA string of the i-th vertex	p_i	point group of the i-th vertex
X	DNA string representing weight	ψ	DNA string representing point group

DNA, deoxyribonucleic acid.

Algorithm 3.3. 1: Generate various routings strands.

(1-1) *Merge*(T_1, T_2)
(1-2) *Annealing*(T_1);
(1-3) *Denaturation*(T_1);
(1-4) *Separation*(T_1,{#$A_1p_1B_1$}, T_4);
(1-5) *Discard*(T_1);
(1-6) *Separation*(T_4,{$A_1p_1B_1$#}, T_5);
(1-7) *Discard*(T_4).

We illustrate the process of the algorithm by taking the GTSP in Figure 3 (starting and ending node is v_1) as an example. Then,

$$T_1 = \{\#A_11B_1, A_21B_2, A_31B_3, A_42B_4, A_52B_5, A_62B_6, A_73B_7, A_83B_8, A_11B_1\#\};$$

$$T_2 = \{\overline{B_1A_2}, \overline{B_1A_3}, \overline{B_1A_4}, \overline{B_1A_6}, \overline{B_1A_7}, \overline{B_1A_8}, \overline{B_2A_1}, \overline{B_2A_6}, \overline{B_2A_7}, \overline{B_3A_1}, \overline{B_3A_5}, \overline{B_3A_8}, \overline{B_4A_1}, \overline{B_4A_5}, \overline{B_5A_3}, \overline{B_5A_4}, \overline{B_5A_6}, \overline{B_5A_8}, \overline{B_6A_1}, \overline{B_6A_2}, \overline{B_6A_5}, \overline{B_6A_7}, \overline{B_6A_8}, \overline{B_7A_1}, \overline{B_7A_2}, \overline{B_7A_6}, \overline{B_7A_8}, \overline{B_8A_1}, \overline{B_8A_3}, \overline{B_8A_5}, \overline{B_8A_6}, \overline{B_8A_7}\};$$

After the above seven steps, the single chain in tube T_5 will encode all the routes from v_1 to v_1. For Figure 3, we have the DNA strands #$A_11B_1A_73B_7A_83B_8A_11B_1$# representing the route $v_1 \to v_7 \to v_8 \to v_1$. This route goes through three vertices, but it doesn't go through all the point groups, so it doesn't satisfy the problem constraint. All we need are DNA strands that represent going through all of the points, so we have to remove the unqualified strands. Since each of the above operations executes in $O(1)$ time [39–43], this step can be finished in $O(1)$.

3.4.2. Delete routes that do not pass through the point group

There is a limit to the feasible route of GTSP, and at least one vertex in each point group is passed. In other words, if a route does not contain any point group information, then it can't be a viable solution. We remove the unqualified chain by searching the feasible chains.

Algorithm 3.3. 2: Select the route passing through each point group at least once.

For $p = 1$ to $p = m$

(2-1) *Separation*(T_5, p, T_6);
(2-2) *Discard*(T_5);
(2-3) *Copy*(T_6, T_5).
(2-4) *Discard*(T_6).

End for

After this step, we delete the route chains that do not go through all the point groups. We consider in the Figure 3, which the value of m is equal to 3, so the DNA strand #$A_11B_1A_21B_2A_73B_7A_83B_8A_62B_6A_11B_1$# (representing the route of transportation $v_1 \to v_2 \to v_7 \to v_8 \to v_6 \to v_1$) can be retained for passing 3 point groups. This operation uses one "For" clause, thus this step can be finished in $O(m)$ since every single manipulation above works in $O(1)$ time [39–43].

3.4.3. Delete chains that pass through a vertex multiple times

We need to compare the weights of different routes and fill in the vertices in the graph that the route does not pass through. Since we require each vertex to be experienced at most once, we attach chains of information that represent vertices that the route does not pass through. If the length of the chain exceeds a certain length, this means that the corresponding chain represents a route that passes through a vertex multiple times, so the sum of the vertices required for each route should not exceed this limit.

Algorithm 3.3. 3: Add vertices information that the route does not pass.

For $j = 2$ to $j = n$

(3-1) *Separation*($T_5, A_jp_jB_j, T_7$);
(3-2) *Append* − *head*($T_5, \psi\psi\psi$);
(3-3) *Merge*(T_5, T_7).

End for

(3-4) *Selection*($T_5, (3n + 5)t, T_8$).

After the above operations, we add the unpassed vertex information to the feasible chain. In Figure 3, we get the number of vertices is eight, and the chain length limit is 29t. Such as the singled strand

$$\#A_11B_1A_21B_2A_73B_7A_83B_8A_62B_6A_11B_1\#\underbrace{\psi\psi\psi\psi\psi\psi\psi\psi}_{3*3}$$

is in the tube T_8 after the step. This operation uses one "For" clause, thus this step can be finished in $O(n)$ since each single manipulation above works in $O(1)$ time [39–43].

3.4.4. Add weight chains for different routes

To select the optimal route, we add the weight chains of corresponding routes to the end of DNA strands. We need to check the weight of the feasible route and then attach the X-chain to the end of the routing chain for comparison.

Algorithm 3.3. 4: Add weight chains that the route pass.

For $i = 1$ to $i = n$ For $j = 1$ to $j = n$

(4-1) *Separation*(T_8, B_jA_i, T_9);
(4-2-1) *If*($T_9 \neq \emptyset$)
(4-2-2) *Append*(T_9, $\underbrace{XX\cdots X}_{Number:\ w_{ij}}$);
(4-2-3) *Merge*(T_8, T_9);
Else
(4-3) *Continue*.

End for
End for

In Figure 3, after adding the weight chains, the route $v_1 \to v_2 \to v_7 \to v_6 \to v_1$ is expressed as:

$$\#A_11B_1A_21B_2A_73B_7A_62B_6A_11B_1\#\psi\psi\psi\psi\psi\psi\psi\psi\ \underbrace{XXX}_{W_{12}}\ \underbrace{XXXX}_{W_{27}}\ \underbrace{XX}_{W_{61}}\ \underbrace{X}_{W_{76}}.$$

Meanwhile, we use two "For" clauses, thus this operation can be finished in $O(n^2)$.

3.4.5. Get the optimal solution chain

The optimal feasible solution chain refers to the minimum sum of edge weights. In the last test tube, we searched for the shortest DNA strands to represent the results of the GTSP. This operation works in O(1) steps.

Algorithm 3.3. 5: Search the optimal solutions.

(5-1) $Sort(T_8, T_{10}, T_{11})$;
(5-2) $Read(T_{10})$.

We can get the shortest DNA strands for the Figure 3:

$$\#A_1 1 B_1 A_7 3 B_7 A_6 2 B_6 A_1 1 B_1 \# \underbrace{\psi\psi\psi\psi\psi\psi\psi\psi\psi\psi\psi\psi\psi\psi\psi}_{15} \underbrace{XX}_{W_{17}} \underbrace{XX}_{W_{61}} \underbrace{X}_{W_{76}}$$

$$\#A_1 1 B_1 A_6 2 B_6 A_7 3 B_7 A_1 1 B_1 \# \underbrace{\psi\psi\psi\psi\psi\psi\psi\psi\psi\psi\psi\psi\psi\psi\psi}_{15} \underbrace{XX}_{W_{17}} \underbrace{XX}_{W_{61}} \underbrace{X}_{W_{76}}.$$

So, the shortest route is either $v_1 \to v_7 \to v_6 \to v_1$ or $v_1 \to v_6 \to v_7 \to v_1$.

4. CORRECTNESS AND COMPUTING COMPLEXITY OF PROPOSED DNA ALGORITHMS

We demonstrate that the algorithm can obtain the solution of GTSP with $O(n^2)$ steps by DNA molecular operations.

Theorem 1. *The GTSP can be solved by the proposed DNA algorithms.*

Proof. We use different DNA biological chains to represent different vertices, point groups and weights to synthesize complete single strands. By eliminating the unsatisfactory chains in all possible paths, all possible DNA chains are obtained, and then through corresponding DNA operations, the solution to the GTSP is obtained. Specifically, we divided it into five steps: Step 1 generates a set T_5 containing all possible solutions of GTSP. Through a series of DNA operations in Step 2, the chains in the set that do not pass through all the groups of points will be deleted. By Step 3, we attached the strand $\psi\psi\psi$ to the DNA chains tail to ensure that the lengths of the strands are the same for the different routes. In Step 4, we attach profit chain X to the end of the corresponding chain to represent the weight value of the passing route. We read out the optimal solution that meets the requirements of the problem in Step 5.

Theorem 2. *Using DNA molecular algorithms, GTSP can be solved in $O(n^2)$ time complexity level.*

Proof. The complexity of each biological operation is within $O(1)$ time [39–43]. Algorithms 3.3.1 and 3.3.5 take $O(1)$ time complexity. Due to the $m < n$, the Algorithm 3.3.2 takes time complexity is less than $O(n)$. Algorithms 3.3.3 and 3.3.4 take $O(n)$ and $O(n^2)$ time complexity respectively. The time complexity T of the algorithm is as follows:

Hence, the algorithms solve the GTSP in $O(n^2)$ time complexity level.

5. SIMULATION EXPERIMENT OF DNA ALGORITHMS

In order to prove the feasibility of the algorithm, the simulation experiment is an indispensable part. Our simulation experiments are mainly carried out in three aspects. One is to reasonably design the DNA representation of each element in the problem, which plays a decisive role in the accuracy of the experiment. The second is to reduce the hybridization error rate in the experimental steps to ensure that the corresponding information can be read quickly and accurately. Finally, the process and results of the experiment are presented.

The calculation of DNA depends on the accuracy of the operation of biochemical molecules; otherwise, it will lead to the accumulation and expansion of errors in biochemical reactions. Meanwhile, DNA sequences that may encourage unexpected probe library hybridization should be excluded. Therefore, designing a suitable DNA sequence is a necessary basis for ensuring accuracy with DNA calculations. To achieve these objectives, seven constraints for DNA sequence design were proposed [44]. These limitations include that long homopolymer chains may have unusual secondary structures; if there is no long homopolymer beam, the melting temperature of the probe library mixture will be more uniform; the probe will only bind weakly where it is not intended to bind; and the affinity of the library chain to itself is very low, etc. To this end, we used the sequence design methods in Ref. [44].

In this paper, we use the computational molecular biology tool Biopython as the system development platform, and Braich's program to generate "appropriate DNA sequences" suitable for biological computing algorithm, which can be used to solve the GTSP. Because the original Braich's program did not find the source code of two functions srante48() and drand48(), the standard function srand() was used instead of function srand48() and the source code of function drand48() was added. Our modified program is used to construct a random sequence of 4 bases for each bit of the library, and to check whether the library chain meets the seven constraints of DNA sequence [44]. If the produced DNA sequence does not meet the restrictions, the program will continue to generate another new sequence. When these restrictions can be met, the sequence is accepted for subsequent biochemical reactions. Using this method, we can get the "appropriate DNA sequences" which meets the restriction conditions to improve the accuracy of biochemical reactions [45–48].

Therefore, taking the GTSP (Figure 3) as an example, the program generates random four-base sequences to form A_i, B_i, #, X and ψ as shown in Table 2. Table 3 demonstrates the DNA node sequence composed by improved Braich's methods.

In Table 4, we also calculate the enthalpy, entropy and free energy of the binding of each probe to the corresponding region in the chains. Their average deviation and standard deviation levels are also shown in Table 4. Routes that pass through no more than two vertices of each point group in Figure 3 are shown in Table 5 (due to there are too many possible routes, we only represent some of them). In the simulation experiment, we derive the optimal solution of the GTSP through the composition structure of the last selected DNA sequences in Table 6.

Table 2 Sequences chosen to represent A_i, B_i, p_i, #, X and ψ ($i \in \{1, 2, \cdots, 8\}$) for the generalized traveling salesman problem (GTSP) in Figure 3.

Bit	$3' - 5'$ DNA Sequence	Bit	$3' - 5'$ DNA Sequence
A_1	$ACAT$	B_1	$GTCC$
A_2	$CTAT$	B_2	$GGCT$
A_3	$TGTC$	B_3	$ATCC$
A_4	$GAAT$	B_4	$ACTG$
A_5	$AATC$	B_5	$CCGT$
A_6	$GCTA$	B_6	$CTTC$
A_7	$GCGT$	B_7	$AGCC$
A_8	$TTGT$	B_8	$TAGT$
1	$TAAG$	2	$CTGA$
3	$AGTC$	#	$CGCT$
X	$CAGC$	ψ	$TGCT$

Table 3 Sequences chosen to represent the elements $A_i p_i B_i$ ($i \in \{1, 2, \cdots, 8\}$) for the generalized traveling salesman problem (GTSP) in Figure 3.

Bit	$3' - 5'$ DNA Sequence	Bit	$3' - 5'$ DNA Sequence
$A_1 1 B_1$	$ACATTAAGGTCC$	$A_2 1 B_2$	$CTATTAAGGGCT$
$A_3 1 B_3$	$TGTCTAAGATCC$	$A_4 2 B_4$	$GAATCTGAACTG$
$A_5 2 B_5$	$AATCCTGACCGT$	$A_6 2 B_6$	$GCTACTGACTTC$
$A_7 3 B_7$	$GCGTAGTCAGCC$	$A_8 3 B_8$	$TTGTAGTCTAGT$

Table 4 The energies over all probe/library strand interactions.

Vertex	Enthalpy Energy H	Entropy Energy S	Free Energy G
$A_1 1 B_1$	110.5	287.7	24.6
$A_2 1 B_2$	95.1	242.9	25.1
$A_3 1 B_3$	103.3	272.1	23.6
$A_4 2 B_4$	98.4	247.3	23.3
$A_5 2 B_5$	109.6	281.1	25.9
$A_6 2 B_6$	105.2	276.2	23.1
$A_7 3 B_7$	104.5	271.9	23.2
$A_8 3 B_8$	99.7	251.3	23.1
Average	103.288	266.313	23.989
Standard deviation	5.356	16.791	1.075

6. CONCLUSIONS

This paper presents DNA biological algorithms for solving GTSP based on the Adleman–Lipton model. In this process, the biological operations are used to produce combined results and to perform screening solutions. Advantages of using DNA algorithms are as follows: first, the algorithms are based on DNA molecules with strong parallel computing power, large storage capacity and low hybridization error rate as we use the specific reasonable coding DNA sequences. Secondly, the proposed algorithms can solve the GTSP at $O(n^2)$ time complexity level. At present, there are few methods to solve the second kind of GTSP, and most of them have the disadvantage of high computational complexity. For example, aiming at the second type of GTSP, Zhao [16] proposed a new GA for adding virtual vertices to generalized chromosomes, that the second kind of GTSP can be solved by the virtual vertex algorithm after a computation transformation. In 2013, Tan *et al.* [49] transformed the GTSP of the second type into the GTSP of the first type by reconstructing the algorithm of the distance matrix. They then used the HCGA algorithm to solve the transformed GTSP of the first type, thus indirectly solved the original problem. These two methods are the second type of GTSP transformation in the solution. In comparison, the algorithm proposed by the Adleman–Lipton model does not require transformation, and it is easier to understand and operate directly. Ant colony optimization algorithm [20] introduces individual variation change route selection. However, as this algorithm is a typical probability algorithm, parameter setting may affect the search time. Garg *et al.* [50] proposed a famous approximate algorithm for GTSP with n vertices and m point groups to generate approximate factor $O(log^2(n)log(log(n))log(m))$. Bontoux *et al.* [51] addresses the solution of the GTSP using a dynamic programming algorithm with the $O(n^2m)$ time complexity. Pintea *et al.* [52] presented an effective meta-heuristic algorithm for solving the problem with the $O((p-1)!\,(nm + nlogn))$ time complexity. In this paper, this algorithm is compared with other heuristic algorithms, and the results of the test problem are given (Table 7). Limited by the current biological computer simulation technology, DNA computing can not be compared by computational simulation. However, our DNA algorithm has many other advantages, such as generality, low energy consumption, efficiency, reducing time and space complexity and fault tolerance. The application potential of DNA molecular computing technology is huge. However, there are still many practical problems to be solved in the research of DNA Algorithm for different problems. Firstly, the complexity of the algorithm implementation, especially the coding ability, may be limited to a certain extent. Secondly, due to the instability of biochemical reaction, there may be some missed solutions, false solutions and wrong solutions. Our results are based on a theoretical model, which presents a DNA strand coding method, simulates DNA experiments and solves the GTSP. Each of the DNA operation used has been implemented at the laboratory level so that the method can be applied in practice. For complex NP problems, the algorithm can also be solved, but the experiment will be more complex. In previous studies, many researchers have used the same perspective to analyze the complexity of DNA computing to solve different problems [53–56]. It needs to develop more powerful multifunctional DNA computing systems. Most DNA computing models can only address a specific type of problem, and the lack of a universal computing system hinders the large-scale promotion of DNA computing. This will lead us to more in-depth research and more challenging development in the field of biotechnology.

In recent years, the research of DNA computing has always been a hot spot. Although there are still some difficulties to be broken through, it is undeniable that DNA has the advantages of storage capability, parallel capability and stability. Further research can consider applying the proposed algorithm to other TSP expansion problems, including mixed Chinese postman problem (MCPP), asymmetric traveling salesman problem (ATSP), generalized covering traveling salesman problem (GCTSP), etc., and trying to establish a unified and standardized model to solve them. We need to explore better ways to combine DNA computing with other computing methods in intelligent systems. We hope that this technological challenge will bring greater progress in science and technology. Moreover, in the development of DNA computing, information recognition, judgment and bio-chain screening can help us to comprehend the origin of computing more deeply, and also promote the process of DNA computing, making it as one of the potential parallel computing methods to solve more complex large data problems.

Table 5 Routes that pass through no more than two vertices of each point group in Figure 3.

Routing	DNA Strands
$v_1 \to v_4 \to v_5 \to v_8 \to v_1$	$3' - CGCTACATTAAGGTCCGAATCTGAACTGAATCC$ $TGACCGTTTGTAGTCTAGTACATTAAGGTCCCGCT - 5'$
$v_1 \to v_4 \to v_5 \to v_8 \to v_3 \to v_1$	$3' - CGCTACATTAAGGTCCGAATCTGAACTGAATCCTGACCGT$ $TTGTAGTCTAGTTGTCTAAGATCCACATTAAGGTCCCGCT - 5'$
$v_1 \to v_4 \to v_5 \to v_8 \to v_7 \to v_1$	$3' - CGCTACATTAAGGTCCGAATCTGAACTGAATCCTGACCGT$ $TTGTAGTCTAGTGCGTAGTCAGCCACATTAAGGTCCCGCT - 5'$
$v_1 \to v_4 \to v_5 \to v_8 \to v_7 \to v_2 \to v_1$	$3' - CGCTACATTAAGGTCCGAATCTGAACTGAATCCTGACCGTTTGTAG$ $TCTAGTGCGTAGTCAGCCCTATTAAGGGCTACATTAAGGTCCCGCT - 5'$
$v_1 \to v_6 \to v_8 \to v_1$	$3' - CGCTACATTAAGGTCCGCTACTGACTTC$ $TTGTAGTCTAGTACATTAAGGTCCCGCT - 5'$
$v_1 \to v_6 \to v_8 \to v_5 \to v_1$	$3' - CGCTACATTAAGGTCCGCTACTGACTTCTTGTAG$ $TCTAGTAATCCTGACCGTACATTAAGGTCCCGCT - 5'$
$v_1 \to v_6 \to v_8 \to v_7 \to v_1$	$3' - CGCTACATTAAGGTCCGCTACTGACTTCTTGTAG$ $TCTAGTGCGTAGTCAGCCACATTAAGGTCCCGCT - 5'$
$v_1 \to v_6 \to v_8 \to v_7 \to v_2 \to v_1$	$3' - CGCTACATTAAGGTCCGCTACTGACTTCTTGTAGTCTAGT$ $GCGTAGTCAGCCCTATTAAGGGCTACATTAAGGTCCCGCT - 5'$
$v_1 \to v_3 \to v_5 \to v_8 \to v_3 \to v_1$	$3' - CGCTACATTAAGGTCCTGTCTAAGATCCAATCCTGACC$ $GTTTGTAGTCTAGTTGTCTAAGATCCACATTAAGGTCC - 5'$
$v_1 \to v_3 \to v_5 \to v_8 \to v_1$	$3' - CGCTACATTAAGGTCCTGTCTAAGATCCAATC$ $CTGACCGTTTGTAGTCTAGTACATTAAGGTCC - 5'$
$v_1 \to v_3 \to v_5 \to v_8 \to v_7 \to v_1$	$3' - CGCTACATTAAGGTCCTGTCTAAGATCCAATCCTGACC$ $GTTTGTAGTCTAGTGCGTAGTCAGCCACATTAAGGTCC - 5'$
$v_1 \to v_2 \to v_6 \to v_8 \to v_7 \to v_2 \to v_1$	$3' - CGCTACATTAAGGTCCCTATTAAGGGCTGCTACTGACTTCTTGTAG$ $TCTAGTGCGTAGTCAGCCCTATTAAGGGCTACATTAAGGTCCCGCT - 5'$
$v_1 \to v_2 \to v_6 \to v_8 \to v_1$	$3' - CGCTACATTAAGGTCCCTATTAAGGGCTGCTACT$ $GACTTCTTGTAGTCTAGTACATTAAGGTCCCGCT - 5'$
$v_1 \to v_2 \to v_6 \to v_8 \to v_7 \to v_1$	$3' - CGCTACATTAAGGTCCCTATTAAGGGCTGCTACTGACTTC$ $TTGTAGTCTAGTGCGTAGTCAGCCACATTAAGGTCCCGCT - 5'$
$v_1 \to v_2 \to v_6 \to v_5 \to v_8 \to v_1$	$3' - CGCTACATTAAGGTCCCTATTAAGGGCTGCTACTGACTTC$ $AATCCTGACCGTTTGTAGTCTAGTACATTAAGGTCCCGCT - 5'$
$v_1 \to v_2 \to v_6 \to v_5 \to v_8 \to v_7 \to v_1$	$3' - CGCTACATTAAGGTCCCTATTAAGGGCTGCTACTGACTTCAATCCT$ $GACCGTTTGTAGTCTAGTGCGTAGTCAGCCACATTAAGGTCCCGCT - 5'$
$v_1 \to v_2 \to v_6 \to v_5 \to v_8 \to v_7 \to v_2 \to v_1$	$3' - CGCTACATTAAGGTCCCTATTAAGGGCTGCTACTG$ $ACTTCAATCCTGACCGTTTGTAGTCTAGTGCGTAG$ $TCAGCCCTATTAAGGGCTACATTAAGGTCCCGCT - 5'$

Table 6 DNA sequences chosen to represent the solutions to the generalized traveling salesman problem (GTSP) in Figure 3.

Routing	DNA Strands
$v_1 \to v_7 \to v_6 \to v_1$	$3' - CGCTACATTAAGGTCCGCGTAGTCAGCCGCTAC$ $TGACTTCACATTAAGGTCCCGCTTGCTTGCTTGCTT$ $GCTTGCTTGCTTGCTTGCTTGCTTGCTTGCTT$ $GCTTGCTTGCTCAGCCAGCCAGCCAGCCAGC - 5'$
$v_1 \to v_6 \to v_7 \to v_1$	$3' - CGCTACATTAAGGTCCGCTACTGACTTCGCGTA$ $GTCAGCCACATTAAGGTCCCGCTTGCTTGCTTGCTT$ $GCTTGCTTGCTTGCTTGCTTGCTTGCTTGCTT$ $GCTTGCTTGCTCAGCCAGCCAGCCAGCCAGC - 5'$

Table 7 Time complexity of different algorithms for generalized traveling salesman problem (GSTP).

Algorithm	Time Complexity
DNA algorithm	$O(n^2)$
Garg *et al.* [50]	$O(log^2(n)log(log(n))log(m))$
Bontoux *et al.* [51]	$O(n^2m)$
Pintea *et al.* [52]	$O((p-1)!\,(nm + nlogn))$

CONFLICTS OF INTEREST

The authors declare that they have no competing interests.

AUTHORS' CONTRIBUTIONS

All authors contributed to the work. All authors read and approved the manuscript.

ACKNOWLEDGMENTS

It is supported by the Open Research Fund of State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, China Institute of Water Resources and Hydropower Research (grant No. IWHR-SKL-201905). The project is also funded by research Start-up project of Wenzhou Business School (grant No. RC202002).

REFERENCES

[1] A.L. Henry-Labordere, The record balancing problem: a dynamic programming solution of a generalized traveling salesman problem, RAIRO. 2 (1969), 43–49.

[2] J.P. Saksena, Mathematical model of scheduling clients through welfare agencies, CORS J. 8 (1970), 185–200.

[3] S.S. Srivastava, S. Kumar, R.C. Garg, P. Sen, Generalized traveling salesman problem through n sets of nodes, CORS J. 7 (1969), 97–101.

[4] G. Laporte, A. Asef-Vaziri, C. Sriskandarajah, Some applications of the generalized traveling salesman problem, J. Oper. Res. Soc. 47 (1996), 1461–1467.

[5] G. Laporte, Y. Nobert, Generalized traveling salesman through n sets of nodes: an integer programming approach, INFOR. 21 (1983), 61–75.

[6] F.R. Qin, Z.H. Luo, P. Dong, Scheduling of underway replenishment for a battle group based on generalized traveling salesman problem, Ordnance Ind. Autom. 37 (2018), 28–31.

[7] G. Laporte, H. Mercure, Y. Nobert, Generalized traveling salesman problem through n sets of nodes: the asymmetrical cases, Discrete. Appl. Math. 18 (1987), 185–197.

[8] M. Fischetti, J.J. Salazar-Gonzalez, P. Toth, A branch-and-cut algorithm for the symmetric generalized traveling salesman problem, Oper. Res. 45 (1997), 378–394.

[9] G.Y. Ning, D.Q. Cao, Y.Q. Zhou, A discrete differential evolution algorithm for TSP problem, Comput. Digit. Eng. 45 (2017), 2136–2142.

[10] B.Y. Cheng, H.Y. Lu, Y. Huang, K.B. Xu, Particle swarm optimization algorithm based on self-adaptive excellence coefficients for solving traveling salesman problem, Comput. Appl. 37 (2017), 750–754.

[11] Y.X. Zhang, X.K. Ding, D.C. Xue, X.T. Wang, An improved ant colony algorithm for traveling salesman problem, Comput. Eng. Sci. 39 (2017), 1576–1580.

[12] C.E. Noon, J.C. Bean, An efficient transformation of the generalized traveling salesman problem, INFOR. 31 (1993), 39–44.

[13] V. Dimitrijevic, Z. Saric, An efficient transformation of the generalized traveling salesman problem into the traveling salesman problem on digraphs, Inf. Sci. 102 (1997), 105–110.

[14] Y. Li, X. Zhao, An effective transformation method of GTSP, J. Inf. Technol. 7 (2013), 169–171.

[15] C.G. Wu, Y.C. Liang, H.P. Lee, C. Lu, Generalized chromosome genetic algorithm for generalized traveling salesman problems and its applications for machining, Phys. Rev. E. 70 (2004), 1–13.

[16] X. Zhao, J.L. Lin, X.Q. Lu, Void vertex genetic algorithm for the second kind of generalized traveling salesman problems, Comput. Eng. Appl. 15 (2006), 78.

[17] M.H. Wang, X.H. Tian, H.X. Liao, Genetic algorithm for optimal design of machining route, Comput. Eng. Appl. 27 (2009), 59–61.

[18] L. Snyder, M. Daskin, A random-key genetic algorithm for the generalized traveling salesman problem, Eur. J. Oper. Res. 174 (2006), 38–53.

[19] Z. Ardalan, S. Karimi, O. Poursabzi, B. Naderi, A novel imperialist competitive algorithm for generalized traveling salesman problems, J. Appl. Soft. Comput. 26 (2015), 546–555.

[20] J.M. Kan, Y. Zhang, Application of an improved ant colony optimization on generalized traveling salesman problem, Energy. Procedia. 17 (2012), 319–325.

[21] S. Smith, F. Imeson, GLNS: an effective large neighborhood search heuristic for the generalized traveling salesman problem, Comput. Oper. Res. 87 (2017), 1–19.

[22] E. Mehdi, A. Belaid, E. Bouazza, A memetic algorithm based on breakout local search for the generalized traveling salesman problem, Appl. Artif. Intell. 34 (2020), 537–549.

[23] L.M. Adleman, Molecular computation of solution to combinatorial problems, Science. 266 (1994), 1021–1024.

[24] R.J. Lipton, DNA solution of HARD computational problems, Science. 268 (1995), 542–545.

[25] Q. Ouyang, P.D. Kaplan, S. Liu, A. Libchaber, DNA solution of the maximal clique problem, Science. 278 (1997), 446–449.

[26] T. Head, G. Rozenberg, R.S. Bladergroen, C.K.D. Breek, P.H.M. Lommerse, H.P. Spaink, Computing with DNA by operating on plasmids, BioSystems. 57 (2000), 87–93.

[27] L. Qian, E. Winfre, Scaling up digital circuit computation with DNA strand displacement cascades, Science. 332 (2011), 1196–1201.

[28] L. Qian, E. Winfree, B. Jehoshua, Neural network computation with DNA strand displacement cascades, Nature. 475 (2011), 368–372.

[29] W. Li, Y. Yang, H. Yan, Three-input majority logic gate and multiple input logic circuit based on DNA strand displacement, Nano. Lett. 13 (2013), 2980–2988.

[30] W.L. Chang, T.T. Ren, M. Feng, Quantum algorithms and mathematical formulations of bio-molecular solutions of the vertex cover problem in the finite-dimensional hilbert space, IEEE Trans. Nanobiosci. 14 (2015), 121–128.

[31] F. Li, J. Liu, Z. Li, DNA computation based on self-assembled nanoparticle probes for 0-1 integer programming problem, Math. Comput. Simulat. 151 (2018), 140–146.

[32] Y. Li, L. Xiao, L. Ruan, Parallel molecular computation of modular-multiplication with two same inputs over finite field GF(2^n) using self-assembly of DNA tiles, Comput. Biol. Chem. 50 (2014), 82–87.

[33] M. Endo, Y. Katsuda, K. Hidak, H. Sugiyama. A versatile DNA nanochip for direct analysis of DNA base-excision repair, Angew. Chem. Int. Edit. 49 (2010), 9412–9416.

[34] S.M. Douglas, I. Bachelet, G.M. Church. A logic-gated nanorobot for targeted transport of molecular payloads, Science. 335 (2012), 831–834.

[35] A. Ogawa, Y. Susak, Multiple-input and visible-output logic gates using signal-converting DNA machines and gold nanoparticle aggregation, Org. Biomol. Chem. 11 (2013), 3272–3276.

[36] X. Zhou, Y. Wang, L.B. Zhong, S.X. Bao, Y. Han, L. Ren, Q.Q. Zhang, Rational design of oriented assembly of gold nanospheres with nanorods by biotin-streptavidin connectors, Nanoscale. 4 (2012), 6256–6259.

[37] W. Deng, J. Xu, Y. Song, H. Zhao, Differential evolution algorithm with wavelet basis function and optimal mutation strategy for complex optimization problem, Appl. Soft Comput. (2020), 106724.

[38] W. Deng, H. Liu, J. Xu, H. Zhao, Y. Song, An improved quantum-inspired differential evolution algorithm for deep belief network, IEEE Trans. Instrum. Meas. 69 (2020), 7319–7327.

[39] K.H. Zimmermann, Z. Ignatova, M.P. Israel, DNA Computing Models, Springer-Verlag, 2008, pp. 146–147.

[40] Z.C. Wang, Z.W. Ji, X.M. Wang, T.H. Wu, W. Huang, A new parallel DNA algorithm to solve the task scheduling problem based on inspired computational model, Biosystems. 162 (2017), 59–65.

[41] Z.W. Ji, Z.C. Wang, T.H. Wu, W. Huang, Solving the 0-1 knapsack problem based on a parallel intelligent molecular computing model system, J. Intell. Fuzzy Syst. 33 (2017), 2719–2726.
[42] X. Zheng, J. Xu, W. Li, Parallel DNA arithmetic operation based on *n*-moduli set, Appl. Math. Comput. 12 (2009), 177–184.
[43] C.A.A. Sanches, N.Y. Soma, A polynomial-time DNA computing solution for the bin-packing problem, Appl. Math. Comput. 215 (2009), 2055–2062.
[44] R.S. Braich, C. Johnson, P.W.K. Rothemund, D. Hwang, N. Chelyapov, L.M. Adleman, Solution of a satisfiability problem on a gel-based DNA computer, Springer, 2001, pp. 27–42.
[45] W.L. Chang, K.W. Lin, J.C. Chen, C.C. Wang, L.C. Lu, M. Guo, M. Ho, Molecular solutions of the RSA public-key cryptosystem on a DNA-based computer, J. Supercomput. 61 (2012), 642–672.
[46] H.Y. Zhang, X.Y. Liu, A CLIQUE algorithm using DNA computing techniques based on closed-circle DNA sequences, Biosystems. 105 (2011), 73–82.
[47] R.S. Braich, C. Johnson, P.W.K. Rothemund, N. Chelyapov, L.M. Adleman, Solution of a 20-variable 3-SAT problem on a DNA computer, Science. 296 (2002), 499–502.
[48] R.B.A. Bakar, J. Watada, W. Pedrycz, DNA approach to solve clustering problem based on a mutual order, Biosystems. 91 (2008), 1–12.
[49] Y. Tan, Z.F. Hao, H. Huang, S. Zhao, Genetic algorithm of distance matrix remodeling for solving the second kind of GTSP, J. South China Univ. Technol. 2013.
[50] N. Garg, G. Konjevod, R. Ravi, A polylogarithmic approximation algorithm for the group steiner tree problem, J. Algorithms. 37 (2000), 66–84.
[51] B. Bontoux, C. Artigues, D. Feillet, A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem, Comput. Oper. Res. 37 (2010), 1844–1852.
[52] C.M. Pintea, P.C. Pop, C. Chira, The generalized traveling salesman problem solved with ant algorithms, Complex Adapt. Syst. Model. 5 (2017), 8.
[53] R. Maazallahi, A. Niknafs, P. Arabkhedri, A polynomial-time DNA computing solution for the N-Queens problem, Procedia Soc. Behav. Sci. 83 (2013), 622–628.
[54] W.X. Li, E.M. Patrikeev, D.M. Xiao, A DNA algorithm for the maximal matching problem, Automat. Rem. Contr. 76 (2015), 1797–1802.
[55] Z.C. Wang, X.M. Ren, Z.W. Ji, W. Huang, T.H. Wu, A novel bio-heuristic computing algorithm to solve the capacitated vehicle routing problem based on AdlemanCLipton model, BioSystems. 184 (2019), 103997.
[56] J. Xu, X.L. Qiang, K. Zhang, C. Zhang, J. Yang, A DNA computing model for the graph vertex coloring problem based on a probe graph, Engineering. 4 (2018), 61–77.