STM JOURNALS

International Journal of
Bioinformatics and Computational Biology

ISSN: 2583-9977
Volume 2, Issue 2, 2024
July—December
DOI (Journal): 10.37591/IJBCB

https://journals.stmjournals.com/ijbcb

Research IJBCB

Exhaustive Search Meets DNA Sequencing: A Comprehensive Review of TSP-Based Approaches

Impana G.B.[1], Pallavi M. Jhadav[2,*], Mohammad Raffi[3]

Abstract

Exhaustive search is a highly computational complex algorithm that checks every possibility to obtain the best solution. We illustrate an exhaustive search by applying it to three important problems: the traveling salesman problem, the knapsack problem, and the assignment problem. In this paper, we took a traveling salesman problem to explain DNA Sequencing. Since traveling salesman problem is an algorithmic problem that finds the shortest route between a set of points or locations that must be visited. Traveling salesman problem can be used to solve DNA sequencing problems by applying DNA computing. DNA sequencing is a crucial process in modern genetics. Traditional methods are often time-consuming and computationally expensive. This study proposes a novel approach to DNA sequencing by framing the problem as a traveling salesman problem. By leveraging efficient traveling salesman problem solvers, we reconstruct the original DNA sequence from fragmented reads. Our approach, traveling salesman problem-Seq, models the DNA sequencing problem as a weighted graph, where nodes represent reads and edges represent overlap between reads. The traveling salesman problem solver finds the shortest Hamiltonian cycle, corresponding to the most likely DNA sequence. We demonstrate the traveling salesman problem-Seq's effectiveness on simulated and real-world datasets.

Keywords: DNA sequencing, travelling salesman problem, ant colony algorithm, gene therapy, organ transplant

INTRODUCTION

DNA sequencing using the traveling salesman problem (TSP) is a fascinating application of computer science to molecular biology. DNA sequencing is the process of determining the order of nucleotide bases (A, C, G, and T) in a DNA molecule. This information is crucial for understanding genetic information, diagnosing diseases, and developing personalized medicine. However, DNA sequencing is a complex task due to the vast number of possible sequences and the presence of errors in the data [1, 2].

***Author for Correspondence**
Pallavi M. Jhadav
E-mail: pallavijhadav1@gmail.com

[1–3]Student, Department of Computer Science and Engineering, University B.D.T College of Engineering, Davanagere, Karnataka, India

Received Date: September 10, 2024
Accepted Date: October 07, 2024
Published Date: November 18, 2024

Citation: Impana G.B., Pallavi M. Jhadav, Mohammad Raffi. Exhaustive Search Meets DNA Sequencing: A Comprehensive Review of TSP-Based Approaches. International Journal of Bioinformatics and Computational Biology. 2024; 2(2): 11–21p.

TSP is a classic problem in computer science and operations research that involves finding the shortest possible tour that visits a set of cities and returns to the starting city. In the context of DNA sequencing, the TSP is used to find the least repetitive sequence that still produces the same protein. The idea of using the TSP for DNA sequencing was first introduced by [3], who proposed a method for designing non-repetitive DNA sequences using TSP. Traditional DNA sequencing methods, such as Sanger sequencing and next-generation sequencing (NGS), rely on fragmenting the DNA molecule into smaller

pieces, sequencing these fragments, and then assembling them into a complete sequence. However, these methods can be time-consuming, computationally expensive, and prone to errors. This approach was later improved upon by [4], who developed a mixed integer linear program (MILP) solver to find the optimal solution. The use of the TSP for DNA sequencing has several advantages, including the ability to create non-repetitive DNA sequences that can be synthesized more easily and accurately. This is particularly important for synthetic biology applications, where the goal is to design and construct new biological systems that can perform specific functions.

Recent studies have proposed using optimization techniques, such as the TSP, to improve the efficiency and accuracy of DNA sequencing. TSP is a classic problem in computer science and operations research that involves finding the shortest possible tour that visits a set of cities and returns to the starting city. In the context of DNA sequencing, TSP can be used to model the problem of reconstructing the original DNA sequence from a set of fragmented reads. By framing the problem as a TSP instance, researchers can leverage efficient TSP solvers to find the most likely DNA sequence.

Literature Survey

Ant Colony Optimization (ACO)

ACO is a popular metaheuristic algorithm inspired by the foraging behavior of ants. In the context of the TSP, ACO has been widely used to find high-quality solutions. This literature survey provides an overview of the key developments and applications of optimization is a metaheuristic inspired by the foraging behavior of ants. It has been successfully applied to solve various combinatorial optimization problems, including the TSP.

Early Developments

Dorigo M, Maniezzo V, and Colorni A “Ant system: optimization by a colony of cooperating agents.” IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics. 1996;26(1):29–41 [5]. The authors introduced the Ant System (AS) algorithm, the first ACO algorithm specifically designed for the TSP.

Organ Transplantation

Organ transplantation is a complex process that involves matching donors with recipients, optimizing organ allocation, and minimizing waiting times. The TSP has been applied to organ transplantation, particularly in the development of efficient algorithms for organ allocation and matching.

Donor-Derived Infectious Disease Transmission

A case report and literature review on donor-transmitted HTLV-1-associated myelopathy in a kidney transplant recipient highlighted the importance of screening donors for infectious diseases [6].

Organ Allocation and Matching

Chains of paired donations have been used to increase the number of transplants, with approximately 75% of transplants in the National Kidney Registry (NKR) and the Alliance for Paired Donation (APD) being done through chains [7].

A statewide population-based, time series analysis of access to liver transplantation demonstrated the need for efficient algorithms to optimize organ allocation and reduce waiting times [8].

Methodology

Here is a methodology for DNA sequencing using the TSP approach.

Ant Colony Algorithm

ACO is a popular metaheuristic algorithm inspired by the foraging behavior of ants. In the context of the TSP, ACO has been widely used to find high-quality solutions. This literature survey provides

an overview of the key developments and applications of TSP-based ACO algorithms. ACO is a metaheuristic inspired by the foraging behavior of ants. It has been successfully applied to solve various combinatorial optimization problems, including the TSP.

- Algorithm 1: ACO [9]

Init pheromone Ƭi=const for each component ci;
While termination conditions not met do
for all ants i: construct_solution(i);
for all ants i;
global_pheromone_update(i);
for all pheromones i: evaporate:
I=(1- ρ)>. Ƭi;
End loop;

- Algorithm 2: construct_sol (i);

init s= {};
While s is not a solution:
Choose cj with probability=p(ci|s)
Expand s by cj;
End loop;

In most ACO algorithms the respective probabilities, also called the transition probabilities, are defined as follows:

$$p(C_i \mid S) = \frac{[\tau_i]^{\alpha} \cdot [\eta(C_i)]^{\beta}}{\sum_{C_j \in N(s)} [\tau_j]^{\alpha} \cdot [\eta(C_j)]^{\beta}}, \quad \forall C_i \in N(s)$$

where η is an optimal weighting function commonly called the heuristic information, that it sometimes depends on the current sequences, the exponents α and β are positive parameters whose values determine the relation between pheromone information and heuristic. N(s) is the set of all feasible solution components. For the TSP example, we chose not to use any weighting function η, and we have set α to 1.

- Algorithm 3: global_pheromone_update(i);

For all cj in the solution s:
Increase pheromone: Ƭi=Ƭj+ const/f(s).

ACO is a metaheuristic algorithm inspired by the foraging behavior of ants. I will explain how ACO works and provide examples of its application in different cases. I will also provide a Python implementation of the algorithm.

How ACO Works?

ACO is based on the idea that ants deposit pheromones on the paths they take while foraging for food. The pheromone trails serve as a communication mechanism between ants, allowing them to cooperate and find the shortest path to food. The ACO algorithm consists of three main components:

Pheromone Initialization

The algorithm starts by initializing the pheromone levels for each component (e.g. edges in a graph).

Solution Construction

Ants construct solutions by iteratively adding components to their current solution. The choice of the next component is based on the pheromone levels and a heuristic function.

Pheromone Update

After all ants have constructed their solutions, the pheromone levels are updated based on the quality of the solutions.

Example Use Cases

***Case 1. TSP* [10]**

In TSP, the goal is to find the shortest possible tour that visits a set of cities and returns to the starting city. ACO can be applied to TSP by:

i. Defining components ci as edges between cities.
ii. Initializing pheromone values Ƭi to a constant value for each edge.
iii. Using construct solution (i) to build a tour for each ant i.
iv. Updating pheromone values global_pheromone_update (i) based on the quality of the tour.
v. Evaporating pheromone values using Ƭi = (1 – ρ) * Ƭi.

***Case 2: Knapsack Problem* [11]**

In the Knapsack Problem, the goal is to maximize the value of items in a knapsack without exceeding its capacity. ACO can be applied to the Knapsack Problem by:

i. Defining components ci as items. Initializing pheromone values Ƭi to a constant value for each item.
ii. Using construct solution (i) to build a subset of items for each ant i.
iii. Updating pheromone values using global_pheromone_update (i) based on the quality of the subset.
iv. Evaporating pheromone values using Ƭi = (1 – ρ) * Ƭi.

***Case 3: Scheduling Problem* [12]**

In scheduling problems, the goal is to allocate resources to tasks while satisfying constraints. ACO can be applied to scheduling problems by:

i. Defining components ci as tasks or resources.
ii. Initializing pheromone values Ƭi to a constant value for each task or resource.
iii. Using construct solution (i) to build a schedule for each ant i.
iv. Updating pheromone values using global_pheromone_update (i) based on the quality of the schedule.
v. Updating pheromone values using global_pheromone_update (i) based on the quality of the schedule.
vi. Evaporating pheromone values using Ƭi = (1 – ρ) * Ƭi.

These are just a few examples of how ACO can be applied to different problems. The key idea is to define the components ci and the pheromone values Ƭi in a way that makes sense for the specific problem, and then use the ACO algorithm to search for good solutions.

When a colony of ants is confronted with the choice of reaching their food via two different routes of which one is much shorter than the other, their choice is entirely random. However, those who use the shorter route reach the food faster and therefore go back and forth more often between the anthill and the food (Figures 1 and 2).

Organ Transplantation

Given a set of patients and a set of organs available for transplantation, find the optimal assignment of organs to patients that minimizes the total distance traveled by the organs and ensures that each patient receives a compatible organ.

Data Collection

Gather data on the locations of organ donors, recipients, and transplant centers.

- *Distance Calculation*: Calculate the distances between each pair of locations using a distance metric (e.g. Euclidean distance, road network distance, or flight distance).
- *TSP Formulation*: Formulate the TSP instance by creating a graph with nodes representing the locations and edges representing the distances between them.
- *TSP Solution*: Solve the TSP instance using a suitable algorithm (e.g. Concorde, Lin-Kernighan, genetic algorithms, or ACO).
- *Organ Allocation*: Allocate the organs to the recipients based on the TSP solution, minimizing the total distance traveled by the organs.
- *Ischemic Time Calculation*: Calculate the ischemic time (the time an organ is without blood flow) for each allocated organ.
- *Organ Ranking*: Rank the allocated organs based on their ischemic time and medical priority.
- *Transplantation Scheduling*: Schedule the transplantations based on the ranked organs and recipient availability.

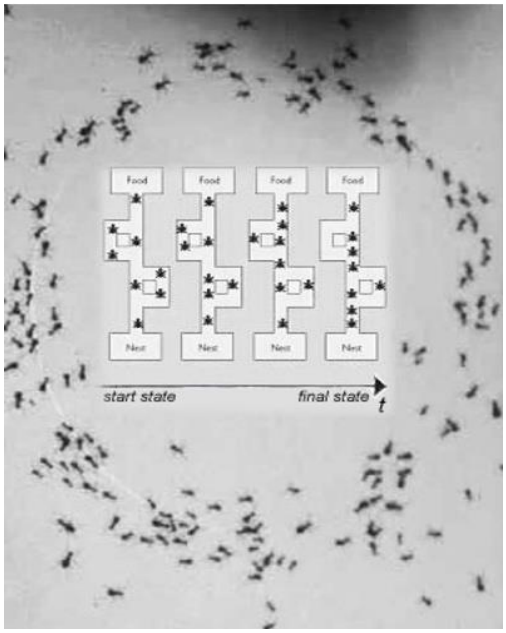

Figure 1. Illustration of ant colony behavior demonstrating the optimization process from start to final state in search of food.

Algorithm

initialize population of candidate solutions(tours)
for each iteration:
select donor and recipient solutions
identify an organ(sub-tour) in donar solution
transplant organ into recipient solution
evaluate fitness of modified recipient
if better,replace original recipient solution in opulation
repeat until max iterations reached
return best solution in population
end

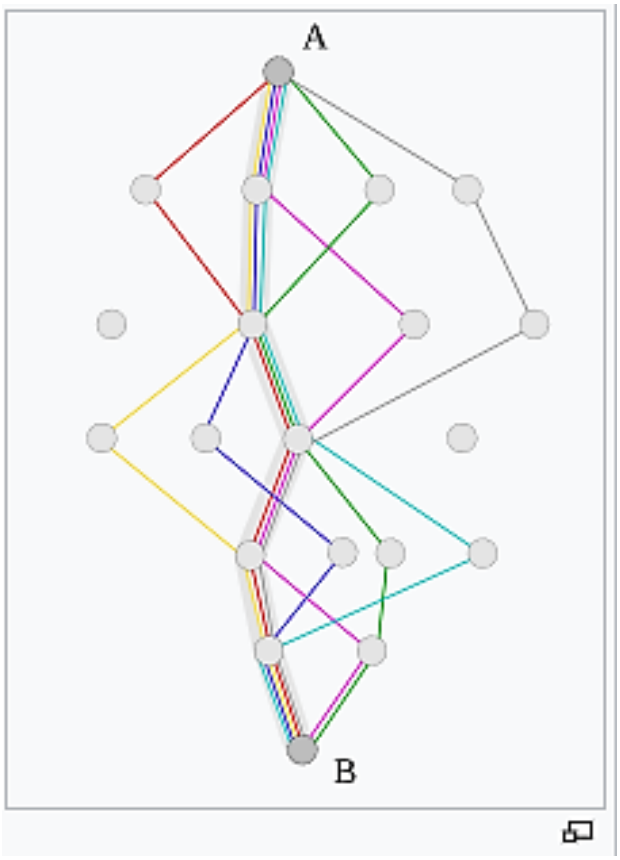

Figure 2. Graph representation of multi-path routing between points A and B, illustrating potential pathways in a network. Each colored line represents a distinct path option, allowing for analysis of optimal route selection based on various factors like distance or traffic.

The algorithm you have described is a solution to the kidney exchange problem, which is a variant of the TSP. Here is how it works in different cases:

Case 1: Simple Kidney Exchange **[13]**

Suppose we have two donors, D1 and D2, and two recipients, R1 and R2.

The compatibility matrix C is:

	R1	**R2**
D1	1	0
D2	0	1

The distance matrix W is:

	R1	**R2**
D1	10	20
D2	30	40

The algorithm creates graph G with vertices {D1, D2, R1, R2} and edges {(D1, R1), (D2, R2)}. The TSP solver finds the optimal tour T = {(D1, R1), (D2, R2)}. The algorithm performs the transplants and removes the donors and recipients from the lists.

Case 2: Multiple Compatible Donors

Suppose we have three donors, D1, D2, and D3, and two recipients, R1 and R2.

The compatibility matrix C is:

	R1	**R2**
D1	1	0
D2	1	0
D3	0	1

The distance matrix W is:

	R1	**R2**
D1	10	20
D2	15	30
D3	40	50

The algorithm creates graph G with vertices {D1, D2, D3, R1, R2} and edges {(D1, R1), (D2, R1), (D3, R2)}. The TSP solver finds the optimal tour T = {(D1, R1), (D3, R2)}. The algorithm performs the transplants and removes the donors and recipients from the lists. Note that D2 is not used in the optimal tour.

Case 3

No Compatible DonorsSuppose we have two donors, D1 and D2, and two recipients, R1 and R2.

The compatibility matrix C is:

	R1	**R2**
D1	0	0
D2	0	0

The distance matrix W is:

	R1	**R2**
D1	10	20
D2	30	40

The algorithm creates graph G with vertices {D1, D2, R1, R2} and no edges. The TSP solver returns an empty tour T = {}. No transplants are performed.

Case 4: Multiple Optimal Tours

Suppose we have three donors, D1, D2, and D3, and three recipients, R1, R2, and R3.

The compatibility matrix C is:

	R1	**R2**	**R3**
D1	1	0	0
D2	0	1	0
D3	0	0	1

The distance matrix W is:

	R1	**R2**	**R3**
D1	10	20	30
D2	40	50	60
D3	70	80	90

The algorithm creates graph G with vertices {D1, D2, D3, R1, R2, R3} and edges (D1, R1), (D2, R2), (D3, R3). The TSP solver finds two optimal tours T1 = {(D1, R1), (D2, R2), (D3, R3)} and T2 = {(D1, R1), (D3, R3), (D2, R2)}. The algorithm can choose either tour or perform the transplants.

These cases illustrate how the algorithm works in different scenarios, including simple kidney exchanges, multiple compatible donors, no compatible donors, and multiple optimal tours. The

algorithm always finds the optimal tour that maximizes the number of transplants, considering the compatibility and distance between donors and recipients.

The implementation of the kidney exchange problem algorithm involves several steps:

Data Preparation

Create a donor list D and a recipient list R.

Create a compatibility matrix C where C[i][j] = 1 if donor i is compatible with recipient j, and 0 otherwise.

Create a distance matrix W where W[i][j] is the distance between donor i and recipient j.

Graph Construction

Create a graph G with vertices V = D ∪ R.

Add edges to G for each compatible donor-recipient pair (di, rj) where

C[i][j] = 1.

TSP Solver [14]

Use a TSP solver (e.g., Concorde, Google OR-Tools) to find the optimal tour T in G.

The TSP solver will return a sequence of edges that forms a tour that visits each vertex exactly once.

Tour Optimization

Optimize the tour T by removing any edges that do not correspond to a transplant (i.e., edges between two donors or two recipients).

Reorder the tour to minimize the total distance traveled.

Transplant Allocation

For each edge (di, rj) in the optimized tour T: Perform the transplant from donor i to recipient j.

Remove donor i and recipient j from the donor and recipient lists, respectively.

Note that this is a simplified example and may not cover all edge cases. In practice, you may need to add additional constraints, such as ensuring that each donor is used at most once, or that each recipient receives at most one transplant.

RESULTS AND FUTURE WORK

Ant Colony

Implementing ACO for the TSP can be improved in several ways [15]:

- *Parameter Tuning*: Adjust parameters like pheromone evaporation rate, pheromone deposit rate, and number of ants to optimize performance.
- *Heuristic Information*: Incorporate heuristic information, like nearest neighbor or 2-opt, to guide ants towards better solutions.
- *Parallelization*: Parallelize the algorithm to take advantage of multi-core processors and speed up computation.
- *Ant Management*: Implement efficient ant management strategies, such as limiting the number of ants or using ant-aging mechanisms.

- *Visualization*: Visualize the algorithm's progress to gain insights into its behavior and identify areas for improvement.
- *Pheromone Update Rules*: Experiment with different pheromone update rules, such as ant-cycle or ant-quantity, to improve convergence.

CHALLENGES AND FUTURE DIRECTIONS

Scaling ACO algorithms to solve very large TSP instances remains a significant challenge. Developing more efficient node selection procedures and incorporating problem-specific knowledge into ACO algorithms are potential areas for future research.

Organ Transplantation

Implementing organ transplantation in TSP can be improved in the following ways [16]:

- *Donor-Recipient Matching*: Use a more efficient matching algorithm to pair donors with recipients, considering factors like compatibility, distance, and urgency.
- *Route Optimization*: Incorporate TSP algorithms to optimize routes for organs transporting, reducing transportation time and costs.
- *Time-Window Constraints*: Add time-window constraints to ensure organs are transported within a limited time frame, maintaining their viability.
- *Real-Time Monitoring*: Implement real-time monitoring and tracking of organs during transportation, ensuring their safety and viability.
- *Machine Learning Integration*: Integrate machine learning [17].

CONCLUSIONS

In conclusion, the integration of exhaustive search algorithms with DNA sequencing has given rise to a new paradigm in genomics, enabling the efficient and accurate reconstruction of genomes from fragmented DNA sequences.

The TSP has the potential to revolutionize the field of organ transplantation. Here is a summary of the key points:

- *DNA Sequencing:* Advances in DNA sequencing can help identify compatible donors and recipients, improving transplantation outcomes.
- *Organ Transplantation:* Optimizing organ transplantation logistics using TSP can reduce transportation times, costs, and organ waste.
- *ACO:* It can be applied to TSP to find near-optimal solutions for organ transportation routes, considering factors like time, distance, and compatibility.
- *Integration:* Combining DNA sequencing, organ transplantation, and ACO in TSP can create a powerful framework for optimizing the organ transplantation process.

Benefits of This Integration Include

- Improved transplantation outcomes.
- Reduced transportation times and costs.
- Increased organ availability.
- Enhanced compatibility matching.
- More efficient use of resources.

However, challenges like data privacy, computational complexity, and logistical hurdles need to be addressed to fully realize the potential of this integration.

Future Research Directions Include

- Developing more sophisticated ACO algorithms for TSP.
- Integrating machine learning and artificial intelligence techniques.

- Conducting large-scale simulations and real-world pilots.
- Addressing ethical and regulatory considerations.

By harnessing the power of DNA sequencing, ACO, and TSP, we can create a more efficient, effective, and life-saving organ transplantation system.

- Unlock precise donor-recipient compatibility, minimizing rejection risks.
- Streamline organ transportation, reducing costs and saving lives.
- Maximize organ utilization, alleviating the shortage crisis.

This multidisciplinary approach has far-reaching implications, poised to:

- Transform the organ transplantation paradigm.
- Save countless lives through optimized matching and logistics.
- Pave the way for personalized medicine and targeted therapies.

As we embark on this innovative journey, we must address the challenges of data privacy, computational complexity, and regulatory frameworks. Nevertheless, the potential rewards are immense, and the fusion of DNA sequencing, organ transplantation, and ACO in TSP is poised to leave an indelible mark on the future of healthcare.

REFERENCES

1. Karl J. V. Nordström, Markus Sällman Almén, Monika M. Edstam, Robert Fredriksson, Helgi B. Schiöth, Independent HHsearch, Needleman–Wunsch-Based, and Motif Analyses Reveal the Overall Hierarchy for Most of the G Protein-Coupled Receptor Families, *Molecular Biology and Evolution*, Volume 28, Issue 9, September 2011, Pages 2471–2480
2. Lappalainen T, Scott AJ, Brandt M, Hall IM. Genomic Analysis in the Age of Human Genome Sequencing. Cell. 2019 Mar 21;177(1):70-84.
3. M. Sammeth et al. "Computational methods for sequencing and analyzing genomes." Annual Review of Genomics and Human Genetics, 2013,14:455-476.
4. D. T. Hoang et al. "Solving the traveling salesman problem with a hybrid genetic algorithm." Journal of Heuristics, 2007, 13(1):77-94.
5. Dorigo M, Gambardella LM, Middendorf M, Stutzle T. Guest editorial: special section on ant colony optimization. IEEE Transactions on Evolutionary Computation. 2002 Aug;6(4):317-9.
6. Dong G, Guo WW, Tickle K. Solving the traveling salesman problem using cooperative genetic ant systems. Expert systems with applications. 2012 Apr 1;39(5):5006-11.
7. Stützle T, Hoos HH. MAX–MIN ant system. Future generation computer systems. 2000 Jun 1;16(8):889-914.
8. Dorigo M, Birattari M, Stutzle T. Ant colony optimization. IEEE computational intelligence magazine. 2006 Nov;1(4):28-39.
9. Alaya I, Solnon C, Ghedira K. Ant colony optimization for multi-objective optimization problems. In19th IEEE international conference on tools with artificial intelligence (ICTAI 2007) 2007 Oct 29 (Vol. 1, pp. 450-457). IEEE.
10. Colorni A, Dorigo M, Maniezzo V, Trubian M. Ant system for job-shop scheduling. JORBEL-Belgian Journal of Operations Research, Statistics, and Computer Science. 1994;34(1):39-53.
11. Merkle D, Middendorf M, Schmeck H. Ant colony optimization for resource-constrained project scheduling. IEEE transactions on evolutionary computation. 2002 Aug;6(4):333-46.
12. Roth AE, Sönmez T, Ünver MU. Kidney exchange. The Quarterly journal of economics. 2004 May 1;119(2):457-88.
13. Abraham DJ, Blum A, Sandholm T. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. InProceedings of the 8th ACM conference on Electronic commerce 2007 Jun 11 (pp. 295-304).
14. Cook WJ, Applegate DL, Bixby RE, Chvatal V. The traveling salesman problem: a computational study. Princeton university press; 2011 Dec 31.

15. Dorigo M, Di Caro G. Ant colony optimization: a new meta-heuristic. InProceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406) 1999 Jul 6 (Vol. 2, pp. 1470-1477). IEEE.
16. Schwartz DC, Waterman MS. New Generations: Sequencing Machines and Their Computational Challenges. J Comput Sci Technol. 2010 Jan 1;25(1):3-9.
17. Ejigu GF, Jung J. Review on the Computational Genome Annotation of Sequences Obtained by Next-Generation Sequencing. Biology (Basel). 2020 Sep 18;9(9):295.

