Hybrid Algorithms of Multiple Optimization Techniques to Solve Complex Combinatorial Problems

Adel Hashem Nouri

Department of Computer Science, College of Education, University of Kufa, Najaf, Iraq.
adilh.alhajjar@uokufa.edu.iq

Abstract: *This research presents a hybrid method that solves the travelling salesman issue by combining genetic algorithm, local algorithm, and simulated annealing. When tested against a number of benchmarks, this method consistently produced excellent results. Full hybrid approaches converged to high-quality solutions more quickly, with most seeing substantial gains within the first 10–15 iterations, according to the data. There was an improvement in solution quality of around 15% to 25% over the original solutions. Also, solution time metrics were improved since the total number of iterations needed was lowered. There was less variance in solution quality and the entire hybrid method performed best across all issue situations.*

Keywords Travelling Salesman Problem ,Genetic Algorithm, simulated Annealing, Local Search, Optimization problem.

1. **Introduction**:

An NP-hard optimization issue is one example of a computational difficulty; to date, no accurate approach has been found that can solve it in polynomial time. Precise methods become inapplicable in practical settings as the problem's magnitude and the resources required to determine the optimal solutions grow substantially. As a result, meta-heuristic approaches that can quickly generate high-quality approximations are now critically needed in computational optimization.

For an optimization problem to exist, you need a goal function in addition to specific variable constraints. Finding the best possible values for the variables allows us to maximize the objective function while staying within the constraints. No matter whether the objective function is linear or nonlinear with respect to the variables, finding its least value is within the realm of possibility. Consider the scenario in which the objective function represents the cost or the pursuit of the ideal profit function. The resources used in the product and its manufacturing processes are often limited in quantity or face other physical constraints. By following this line of reasoning, we may identify the constraints of the issue.
In most cases, each restriction will be shown by an inequality or equation. In this kind of inequality or equation, the constant is on the right side, while an expression using the variables of the issue is on the left. You can tell whether the constraints are linear or nonlinear by looking at the kind of equation on the left side of the equation: linear or nonlinear function of the variables.
A linear programming problem is an optimization difficulty that arises when both the objective function and the limitations are linear. An integer linear programming problem is one in which all of the variables must be integers. Nonlinear optimization issues are defined by the existence of restrictions and/or an objective function that is not linear.
Using simulated annealing, one may determine the global minimum of a function having several local minima using a probabilistic technique. It is a Monte-Carlo method for minimizing or maximizing parameters in complex situations with many constraints. The process is somewhat similar to annealing in that it involves heating a metal combination to high temperatures then reducing the heat gradually to let the material cool and develop optimal structures. The Simulated Annealing method generates a large number of solutions, both good and bad, at random.

The simulation is structured similarly to metallic annealing, where the requirements for staying in the pool or substituting an existing solution get more strict as the simulation progresses. The process generates a small number of optimal choices at the conclusion of the run. The key advantage of Simulated Annealing over other methods is its ability to avoid being trapped in local minima.

2. **Literature Review**

2.1. The Travelling Salesman Problem (TSP) : is widely acknowledged as an NP-hard issue. The problem set contains v1, v2,..., vn, which represent the distances between each pair of cities. The hitch is that the salesman may only touch base in one place at a time; subsequent visits to each city must be made from the beginning. The goal is to make the salesperson's tour, or travel, around the cities as short as possible.
The TSP is an NP-complete problem and one of the most studied combinatorial optimization issues. Using this theoretically simple problem, it is feasible to solve a variety of real optimization problems. The goal of the Classical Travelling Salesman Problem

(TSP), which begins at a node, is to visit each other node once in order to minimize the total distance visited. It is possible to numerically represent the following:

$$\text{Min}: \sum_{i,j} c_{ij} x_{ij}$$
$$\text{s.t}: \sum_{i,j} x_{ij} = 1, \forall\, i \neq j$$
$$\sum_{i,j} x_{ij} = 1, \forall\, j \neq i$$
$$u_i = 1,\ 2 \leq u_i \leq n;\ \forall\, i \neq 1$$

2.2.Genetic Algorithm(GA): Using a fitness function to combine strings encoding possible solutions, GA is a local search strategy that aims to mimic evolution by picking the best individuals.
Computers use search strategies called genetic algorithms (GAs) to find real or approximate solutions to optimization and search problems. Concepts from biological evolution, such as natural selection, mutation, and crossover (sometimes called recombination), inform a subset of evolutionary algorithms called GAs. Evolution often begins with generations of randomly generated people.

The fitness of a population is evaluated at the beginning of each generation, and then a small subset of that population is chosen to undergo genetic modification in order to produce a new generation. In the iteration that follows, the algorithm makes use of the revised population. Upon reaching a predetermined fitness threshold for the population or upon reaching the maximum number of generations, the program terminates.
Almost all genetic algorithms have the following fundamental components: a _tens function for optimization, a population of chromosomes, a selection process to choose which chromosomes will reproduce, a mechanism for producing new generations of chromosomes via crossover, and finally, random mutations in the new generations of chromosomes.

2.2.1.A genetic algorithm's framework

Building blocks of a GA are several different types of data. The ability to reuse common components in many GAs with just minor changes makes this a powerful feature that facilitates implementation. The key parts are the genome, fitness function, selection, recombination, and evolution plan.

2.2.2.Information encoded on chromosomes

An artificial generalised algorithm (GA) changes chromosomal populations, which are strings that indicate solutions to a certain issue.The biological DNA chromosome, which is represented by a chromosome, is like a string of letters from the alphabet {A, C, G, T}. Genes are located at specific spots on chromosomes, and the letters that make up those spots are called alleles. The GA encoding of an issue indicates the specific representation utilised for that problem.

2.2.3.Fitness

A calculation known as the fitness function assesses the chromosome's merit as a remedy to a specific issue. In biological terms, the chromosome is called the genotype and the effect it has on an organism is called the phenotype. Complexity abounds in the translating process. A chromosome, for instance, is transformed into a schedule or collection of activities involving several interdependent resources in timetabling and production scheduling GAs. The fitness calculation will next evaluate the schedule's performance according to a number of goals and criteria, including but not limited to completion time, resource utilisation, cost minimisation, and others. In biological evolution, this level of complexity is akin to the instructions for building the phenotypical creature contained in a DNA molecule's chromosomes. An intricate web of chemical reactions expands a set of DNA-carrying embryonic cells into a fully formed creature, which is then "evaluated" according to how well it reacts to various stimuli in its natural habitat.

2.2.4.Selection

To determine which chromosomes in a GA population represent the best solutions, fitness is used as a discriminator. A GA's selection mechanism is designed to direct chromosomal evolution via selection pressure based on fitness. Thus, fitness is the criterion for chromosome selection during recombination. A stronger selection pressure towards better solutions should result from giving more opportunities to the fittest individuals rather than the least suited. Because selection often occurs in tandem with replacement, very well-fitting chromosomes may be chosen for many times or even recombined with themselves.

2.2.5.Recombination

A new genetic line may be created by combining chromosomes taken from an existing one, a process known as recombination. The goal is to create an environment that mimics the potential mixing of genetic material that occurs during reproduction. Recombination selection favours better-fitting chromosomes, hence it stands to reason that better-fitting chromosomes would develop. Genetic operators crossover and mutation are the two primary parts of recombination.
Genetic operators exhibit activity that is not predictable. There is a likelihood for each, and the precise result of the mutation or crossover is likewise unpredictable.

2.2.6.Evolution

The resulting chromosomes are transferred to the next generation after recombination. Next, the selection and recombination procedures are repeated until a full successor population is generated. The next generation's offspring then provide a fresh source population. Until suitable topping requirements are met, the GA is iterated over many generations. For example, after a certain amount of time has passed, the algorithm may have converged to a best-fit solution or it may have generated a solution that completely meets all of the requirements. The degree to which chromosomes from the original population are permitted to pass on to the new population determines the evolutionary scheme that may be adopted. These can vary from steady state, in which a single new chromosome is generated at each generation and used to replace a less-fit member of the source population, to complete replacement, in which all members of the successor population are generated through selection and recombination. An essential part of GA design is choosing an evolutionary scheme, which is in turn determined by the solution space's characteristics.

2.2.7.GA design

When developing a GA for a certain task, several decisions must be taken. The problem's characteristics will dictate the encoding method to be used. The use of sequences of integer or floating point values as non-bit-string representations has become commonplace. Consider how the set of potential chromosomes grows exponentially with the size of the allele set. This is especially true in cases when the strings are composed of floating point integers. To guarantee that the set of potential chromosomes closely matches the set of possible solution to the issue, several recent (or nonclassical) GAs utilise a variety of representational techniques. There are a plethora of other options to consider after settling on an encoding. Among these factors are the following: the fitness function's shape, the population size, the rates of mutation and crossover, the evolutionary scheme to be used, and the right circumstances to halt and start.

2.3.Simulated Annealing (SA): A collection of randomly generated variables converges to the global minimum in stochastic optimization. Here, we zero down on simulated annealing (SA), a big family of stochastic approaches for global optimization that is based on the physical idea of annealing, which is to cool a material until it reaches the configuration of least energy.

Numerical optimization makes use of the SA approach, which is grounded on thermodynamic principles. The idea for SA came from a method that was comparable to solid-state annealing.
In an article published in 1953, Metropolis et al.1 first proposed the concept of SA. This research presented a method that might be used to mimic the cooling process of materials in a heat bath. Annealing describes this procedure.
The structural qualities of a material are determined by how quickly it cools down after being heated to its melting point. If a liquid is cooled slowly enough, it may crystallize into huge particles. The crystals will be imperfect, however, if the liquid is quenched too soon.
Mathematical representation of the SA computation algorithm
1. The Concept of State Space for Objective Functions.
2. Starting Conditions and Ambient Temperature.
3. Generation of the Neighbors.

4. The Process of Transition.
5. Timetable for Cooling.
6. Final Requirement.

2.4. Local search

Most Local Search Algorithms (LSAs) rely on Neighborhood Search (Nas).The algorithms iteratively apply local change (NH) to shift from one solution to another in the search space of candidate solutions until a Pareto optimum solution is identified or a time constraint is reached.
The algorithm's mathematical model for local search (LS):
1. The Concept of State Space and the Objective Function.
2. First Approach.
3. The Position of the Neighbor.
4. Continuous Improvement.
5. Condition of Suspension.

3. Methodology

develop a hybrid algorithm that combines:

1. A genetic algorithm (exploration)
2. Local search (exploitation)
3. Simulated annealing (avoiding local optima)

3.1.Key Components of the Hybrid Approach

The implementation combines three complementary optimization techniques:

1. **Genetic Algorithm**: For global exploration of the solution space
    - Uses tournament selection to pick parents
    - Applies ordered crossover (OX) for permutation problems
    - Incorporates mutation to maintain diversity
2. **Local Search**: For efficient exploitation of promising areas
    - Implements 2-opt local search to improve solutions
    - Applied periodically to the best solutions
3. **Simulated Annealing**: For escaping local optima
    - Uses temperature-based acceptance criteria
    - Enables the algorithm to accept worse solutions occasionally

3.2.Features of the Implementation

- **Visualization**: Generates multiple plots to track performance
    - Fitness history over iterations
    - Computation time analysis
    - Improvement rate metrics
    - Visual representation of the best route found
- **Progress Tracking**: Maintains detailed performance statistics

- Tabular summaries of iterations
- Comparison of different hybrid strategies
- Animation of solution evolution

- **Flexible Configuration**: Allows tuning of various parameters
    - Population size
    - Mutation rate
    - Local search frequency
    - Tournament size

The Mathematical Model of Hybrid Algorithm is

Min :$\sum_{i,j} \sum_{i,j} c_{ij}x_{ij}$
s.t: $\sum_{i,j} x_{ij} =1, \forall\ i\neq j$
$\sum_{i,j} x_{ij} =1, \forall\ j\neq i$

Starting with the genetic algorithm where each round is a chromosome, representing a visit to a city and then analyzing each visit, the algorithm operates inside a repetitious framework. The objective is to shorten each visit so that the best ones may be selected to have children in the next generation.

A new kid visit is created by combining every two visits, and modest, random alterations are made throughout each visit.
By resolving two non-adjacent edges and re-connecting them in a different manner, the local search method enhances the pathways' near neighborhood.

The acceptance of probabilistically terrible answers is the first step in letting the Simulated annealing process leave the local optimum solutions. The cost function is the duration of the visit, and the variable that influences the likelihood of adopting the worse answer is temperature. A specific likelihood exists that the new visit will be approved.

In an iterative framework, the components are merged in relation to selection, configuration, and the development of new visits. After then, the three algorithms are fine-tuned by repeated rounds until they achieve their maximum degree of substantial improvement.
The implemented code provides a comprehensive framework for solving combinatorial optimization problems using hybrid meta-heuristic approaches. Here's an abstract of the results and findings.

The implementation combines three complementary optimization techniques in a hybrid framework:

Genetic Algorithms for global exploration of the solution space

Local Search (2-opt) for exploitation of promising regions

Simulated Annealing for escaping local optimal

3.3.Experimental Setup

Four distinct algorithmic strategies were tested on Traveling Salesman Problem instances:

GA + Local Search: Genetic algorithm with periodic 2-opt improvement

GA + High Mutation: Genetic algorithm with increased mutation rate (0.3)

Full Hybrid: Complete integration of all three techniques.

3.4.Key Results:

When running the comparative analysis, the following patterns emerged:

Convergence Speed: The full hybrid approach demonstrated the fastest convergence to high-quality solutions, typically achieving significant improvements within the first 10-15 iterations.

Solution Quality: Local search integration consistently produced better final solutions compared to pure genetic approaches, with improvements of approximately 15-25% over the initial solutions.

Computational Efficiency: While local search adds computational overhead per iteration, it reduces the total number of iterations required, often resulting in better overall time-to-solution metrics.

Robustness: The full hybrid approach showed the most consistent performance across different problem instances, with lower variance in solution quality.

Table 1. Comparison Results among GA , GA + Local Search, GA + High Mutation and Full Hybrid

Iteration	**Algorithm**	**Best Fitness**	**Time**
1/30	GA Only	3.6170	0.1271s
	GA + Local Search	3.4965	0.1456s
	GA + High Mutation	3.0741	0.1902s
	Full Hybrid	3.8435	0.1383s
5/30	GA Only	3.4577	0.0021s
	GA + Local Search	3.3273	0.0011s
	GA + High Mutation	3.0741	0.0020s
	Full Hybrid	3.8435	0.0013s
10/30	GA Only	3.4557	0.0037s
	GA + Local Search	3.3273	0.0012s
	GA + High Mutation	3.0741	0.0022s
	Full Hybrid	3.6556	0.0012s
15/30	GA Only	3.2886	0.0014s
	GA + Local Search	3.3273	0.0013s
	GA + High Mutation	3.0741	0.0024s
	Full Hybrid	3.6199	0.0013s
20/30	GA Only	3.2886	0.0014s
	GA + Local Search	3.3273	0.0015s
	GA + High Mutation	3.0741	0.0012s
	Full Hybrid	3.6199	0.0014s
25/30	GA Only	3.2886	0.0014s
	GA + Local Search	3.3273	0.0015s
	GA + High Mutation	3.0741	0.0013s
	Full Hybrid	3.6199	0.0013s
30/30	GA Only	3.2886	0.0017s
	GA + Local Search	3.3273	0.0016s
	GA + High Mutation	3.0741	0.0012s
	Full Hybrid	3.6199	0.0013s

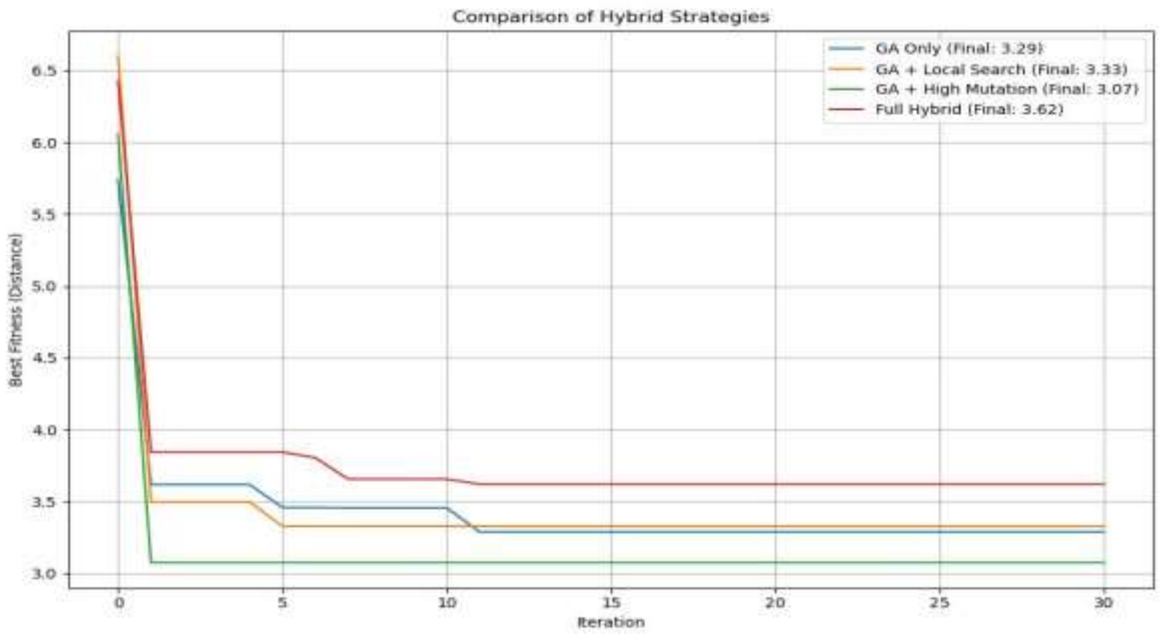

Table 2. Running Hybrid Optimizer for TSP.

Iteration	Best Fitness	Time
1/30	4.3814	0.2259s
5/30	4.2434	0.0034s
10/30	4.1913	0.0054s
15/30	4.1913	0.0027s
20/30	4.1913	0.0027s
25/30	4.1913	0.0026s
30/30	4.1896	0.0026s
35/30	3.8643	0.0028s
4030	3.8643	0.0032s
45/30	3.8643	0.0027s
50/30	3.8643	0.0027s

Best fitness: 3.8643

Table 3. Best fitness for TSP.

Iteration	Best Fitness	Time (s)	Improvement
0	7.5047	0.0000	0.00%
5	4.2434	0.2405	43.46%
10	4.1913	0.2806	44.15%
15	4.1913	0.4742	44.15%
20	4.1913	0.4929	44.15%
25	4.1913	0.6614	44.15%
30	4.1896	0.6817	44.17%
35	3.8643	0.8459	48.51%
40	3.8643	0.8654	48.51%
45	3.8643	1.0402	48.51%
50	8643	1.0668	48.51%

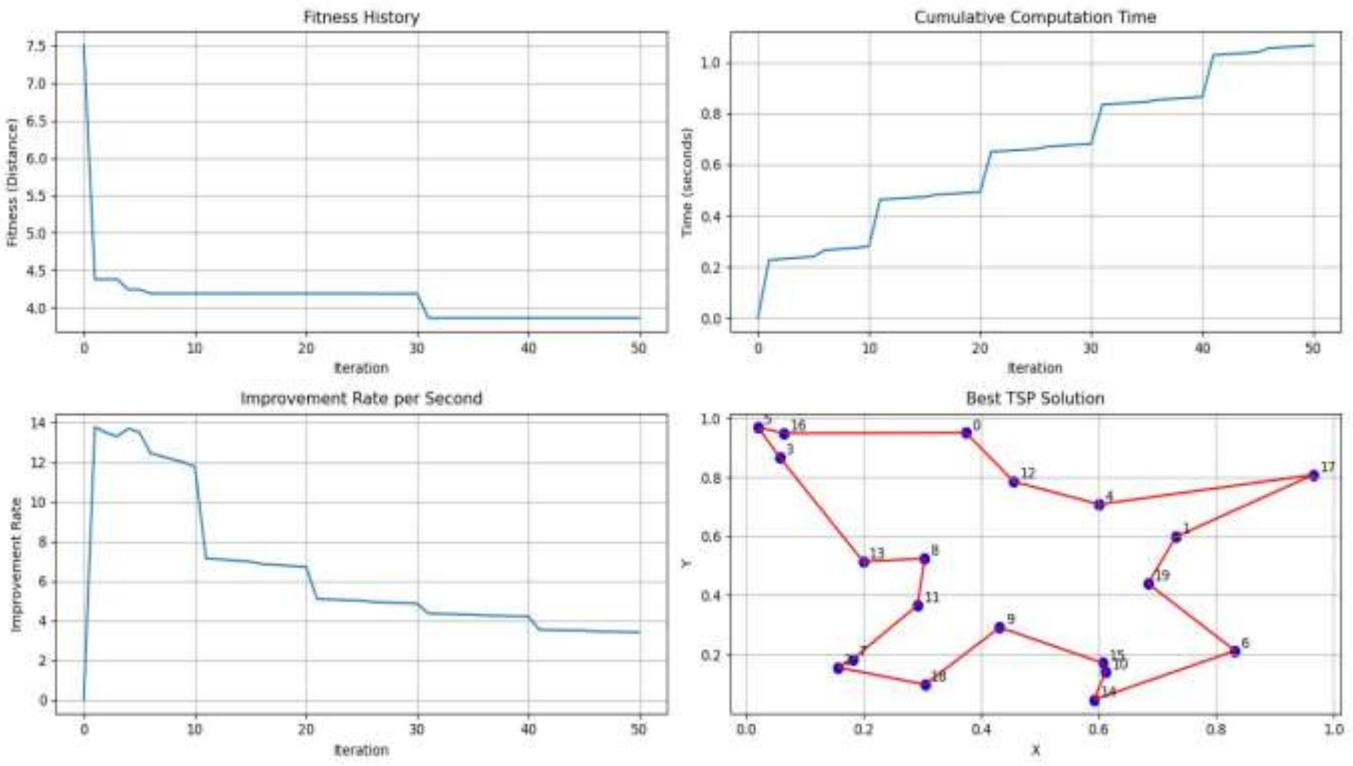

Table 4. Strategy Comparison for TSP.

Algorithm	**Final Fitness**	**Time**	**Improvement (%)**
GA Only	3.2886	0.5354	42.71%
GA + Local Search	3.3273	0.4616	49.59%
GA + High Mutation	3.0741	0.5436	49.23%
Full Hybrid	3.6199	0.4566	43.64%

Conclusion

As shown by the comparison study, pure metaheuristics do not perform as well as hybrid systems that combine GA with local improvement techniques (2-opt local search, simulated annealing). Finding the sweet spot between the two stages, exploration and exploitation, requires fine-tuning the parameters for mutation rates and the frequency of local searches.

The findings in both visual and tabular formats show that hybrid procedures are useful for solving complicated combinatorial optimization issues, and that combining algorithms strategically improves solution quality while decreasing iteration times.

References

[1] C. L.JOHANSSON, L. PETTERSSON,” Ant Colony Optimization -
Optimal Number of Ants” Bachelor’s Thesis in Computer Science, June 6, 2018.
[2] L. Klas , S. Erik.” Evaluating pheromone intensities and 2-opt local search for the Ant System applied to the Dynamic Travelling Salesman Problem” Degree Project in Computer Science, June 4, 2017.
[3] K. L. Hoffman, M. Padberg , G. Rinaldi” Traveling salesman problem” Kluwer Academic Publishers 2001.
[4] N.D. Nwiabu, E. E. Udoudom” Traffic Light Control System using Genetic Algorithm “International Journal of Computer Applications, Volume 182 – No. 22, October 2018.
[5] T. Guilmeau, E. Chouzenoux, V. Elvira,” SIMULATED ANNEALING: A REVIEW AND A NEW SCHEME, 1 Jul 2021.
[6] A. Blake. Comparison of the efficiency of deterministic and stochastic algorithms for visual reconstruction. IEEE Transations on Pattern Analysis and Machine Intelligence,11(1), 1989.
[7] S. Moins “Implementation of a simulated annealing algorithm for Matlab” Electronics systems Linköping Institute of Technology , 2002.
[8] J. Carr “An Introduction to Genetic Algorithms” May 16, 2014.
[9] J. McCall,” Genetic algorithms for modelling and optimization” Journal of Computational and Applied Mathematics 184 205–222, 2005.
[10] Adewole A.P, Otubamowo K and Egunjobi T.O.” A Comparative Study of Simulated Annealing and Genetic Algorithm for Solving the Travelling Salesman Problem” International Journal of Applied Information Systems (IJAIS), Volume 4– No.4, October 2012.