VOL. 19, NO. 24, DECEMBER 2024 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

AN INNOVATIVE METHOD TO ESCAPE LOCAL MINIMA USING QUANTUM SYSTEM DEGENERACY IN SIMULATED ANNEALING

Bouchaib Aylaj[1], Mostafa Belkasmi[2], Said Nouh[3] and Fouad Ayoub[4]
[1,4]Computer Sciences, CRMEF-Rabat, Morocco
[2]ENSIAS, MED5 University, Morocco
[3]Faculty of Sciences BM, H2C University, Morocco
E-Mail: bouchaib_aylaj@yahoo.fr

ABSTRACT

Simulated Annealing (SA) is often used to provide satisfactory solutions to solving combinatorial problems. However, the SA is limited by its tendency to get stuck in local minima, which often prevents the global optimal solution from being reached for large or very complex problems. To overcome this obstacle, we present a new method inspired by the properties of degenerate quantum systems, named Degenerate Quantum Simulated Annealing (SA-DQS). This innovative approach exploits the coexistence of several quantum states of the same energy; this multiplicity of states is advantageous because it allows to finding of several solutions with identical energies, thus increasing the chances of discovering and building a promising solution among them, and of getting out of the traps of local minima more efficiently. The traveling salesman problem (TSP) is used as an application benchmark to validate the effectiveness of SA-DQS. Compared to standard simulated annealing (SA-S), SA-DQS showed superior performance on multiple instances of TSP. On average, the solutions provided by SA-DQS are equal to or closer to the optimum known in TSPLIB, in addition, it also offers a shorter computation time compared to that of other heuristic methods.

Keywords: simulated annealing, TSP, degenerate quantum, combinatorial optimization.

Manuscript Received 16 September 2024; Revised 13 December 2024; Published 25 January 2025

INTRODUCTION

The traveling salesman problem (TSP) is one of the classic challenges of optimization and remains a common benchmark for researchers and computer scientists to measure the quality and effectiveness of their suggested approaches to solving combinatorial problems. The TSP has a high combinatorial complexity, making it difficult to find exact solutions and it's classified as a hard problem in the NP complexity class (C.H. Papadimitriou *et al* 1978).

Since the 1980s, the solution of the TSP has evolved through various algorithmic methods, categorized into two main classes: precise and heuristic. Precise methods offer exact solutions but are often limited to small instances due to their complexity. Dynamic programming (V.B. Lobo *et al* 2016) breaks down the TSP into smaller sub problems and is effective for small instances, but quickly becomes impractical for large ones due to the rapid growth in the number of sub-problems. Branch and Bound optimize by eliminating unpromising branches from the search tree, which performs well for small problems, but is inefficient for large instances (R. Radharamanan *et al* 1986). At the same time, heuristics are more flexible and adapted to large instances of the TSP, although they do not always guarantee an optimal solution. Genetic algorithms (GA), (J. Yang *et al* 2008) inspired by biological evolution, explore the solution space through selection, crossover, and mutation operations, but can get stuck in suboptimal solutions due to premature convergence. Improvements, such as hybrid algorithms (Chun-Wei *et al* 2014) (R. Baraglia *et al* 2001) combining GA with other methods, have been developed to overcome these limitations. Particle swarm optimization (PSO) (B. A. S. Emambocus *et al* 2001) (Y. Cui *et al* 2020) (X.H. Shi *et al* 2007) mimics the social behavior of swarms to explore the search space and shows good performance for medium-sized instances by dividing the global space into particle-managed sub regions. Neural networks (Z. Xing *et al* 2020)(A. H. Gee *et al* 1995), especially through hybrid approaches such as the self-organizing map (SOM) integrated with a scalable algorithm, accelerate convergence toward optimal solutions and can process TSPs of variable size, although they can sometimes offer less precision. Ant colony optimization (ACO) (K. J. man et al 2012)(H. Zhu *et al* 2019)], inspired by ant foraging behavior, is effective in generating optimal solutions, but might only find local optimal solutions. To improve this, strategies such as the introduction of individual variations and distance-pheromone operators have been developed to increase the diversity of solutions. Finally, simulated annealing (SA), (Z. Wang *et al* 2009) (Lalaoui M. *et al* 2018) (S. H Zhan *et al* 2018) which mimics the process of cooling metals to find a low-energy configuration, is often used because of its balance between solution quality and execution time. Recent variants, such as the two-step annealing algorithm (Wang, L. *et al* 2019) (Shi-hua Z. *et al* 2016), increase the diversity of solutions and outperform many other heuristics. Each method, whether precise or metaheuristic,

brings a unique perspective to solving TSP, influenced by the size and specific complexity of the instance.

In this work, we propose an optimization method, named Degenerate Quantum Simulated Annealing (SA-DQS). This innovative approach uses the properties of degenerate quantum systems, where several quantum states can coexist with the same energy, to better explore the space of solutions and escape local minima. Unlike traditional simulated annealing methods that rely on a single processing system, SA-DQS uses two separate processing subsystems. To illustrate the efficacy of this approach, we tested it on the TSP, a classic challenge in combinatorial optimization. This paper continues with the following structure. In the following sections, we provide an introduction to Simulated Annealing and TSP. Section 3 introduces our proposed method, SA-DQS. Section 4 presents the numerical simulation results. Finally, we finish with future perspectives in section 5.

SIMULATED ANNEALING AND TSP

A. Simulated Annealing (SA)

The process of metal annealing in metallurgy serves as the foundation for the idea of SA, developed in the 1950s by (Metropolis *et al.* 1987) seeking to model the cooling of molten metals to achieve an ideal crystalline structure forming such a structure is complex and is influenced by the rate at which metals cool. A starting temperature that is insufficiently high will fail to melt the metals, while a rapidly applied cooling process introduces imperfections to the final solid structure Figure-1. Simulated annealing was introduced into combinatorial optimization in the early 1980s by (S. Kirkpatrick *et al* 1983) and (V. Cerny 1985).

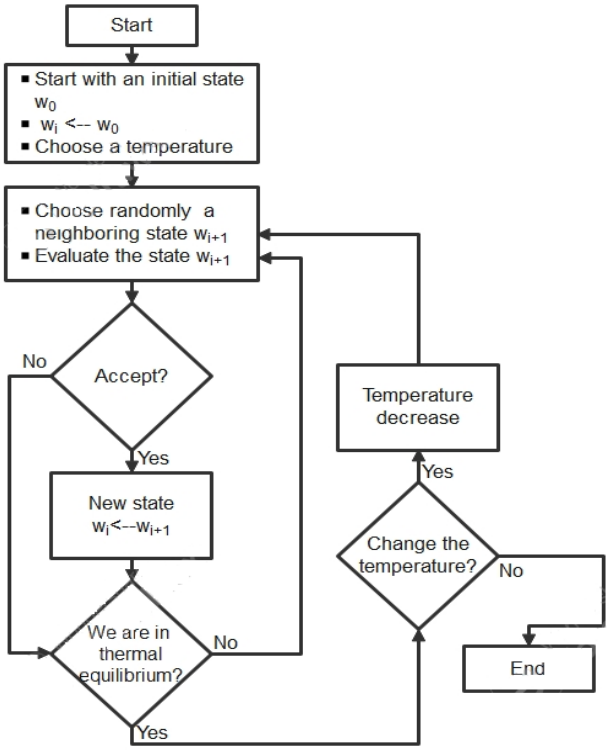

Figure-1. A flowchart of the SA algorithm.

B. Traveling Salesman Problem

The TSP is recognized as a major combinatorial problem, its intrinsic complexity makes it a crucial subject of study for the development of efficient algorithms and resolution methods. The description of the problem is fairly simple: a traveler should visit a given set of cities only once, minimizing the total distance traveled. Formally, the TSP can be formulated as a discrete optimization problem, where the goal is to find the optimal permutation of cities to visit to minimize the total length of the route.

The mathematical cost formulas used to solve the TSP can vary slightly depending on how the problem is formulated and how the data is represented in space. However, the classic cost formulation for TSP is based on minimizing the total distance traveled. Mathematically, this can be expressed as the minimization of the cost function, where the total cost depends on the distance between consecutive cities in the permutation.

Let's have a set of n cities, numbered from 1 to n, and that *would* be the distance between city i and city j. The permutation of cities is represented by a sequence $p=(p_1, p_2, \ldots, p_n)$, where p_i is the number of the visited city in position i, the cost function F(p) for the TSP is usually defined as follows:

$$F(p) = \sum_{i=1}^{n-1} d_{p_i,p_{i+1}} + d_{p_n,p_1} \quad (1)$$

This function represents the sum of distances between cities in the permutation, adding the distance

between the last visited city and the first to complete the cycle.

In the case of Euclidean TSP, where the positions of cities are given in an Euclidean space, the cost formula can be modified to use the coordinates *(x,y)* of Cities. Si (x_i,y_j) are the coordinates of the city *i*, then the distance of Euclidean between two cities is:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (2)$$

Thus, the cost function for the Euclidean TSP becomes:

$$F(p) = \sum_{i=1}^{n-1} \sqrt{\left(x_{p_i} - x_{p_i+1}\right)^2 + \left(y_{p_i} - y_{p_i+1}\right)^2} + \sqrt{(x_{p_n} - x_{p_1})^2 + (y_{p_n} - y_{p_1})^2} \quad (3)$$

This cost function serves as the basis for evaluating the quality of a given solution. Algorithms seek to minimize this cost function to find an optimal solution.

THE PROPOSED SIMULATED ANNEALING

A. Quantum System Degeneracy

The concept of "degeneracy" in the context of quantum systems refers to the situation where different quantum states of a system have the same energy. This means that these states are indistinguishable in terms of energy and therefore can be occupied with the same probability.

Degeneracy can come from different sources, including symmetries of the system, interactions between its components, or conditions imposed on the system (Kardar M. 2007). It can be seen as beneficial as it can provide greater robustness and stability to the system against external perturbations. However, in other cases, it can make solving certain problems more complex, particularly when resolving quantum equations to describe the system (Kardar M. 2007).

The standard simulated annealing has been used for the solidification problem of a liquid metal and is applied to a system where each state corresponds to a single energy (absence of degenerate states). Now, with the use of the degenerate structure of systems, we can have a significant impact on the convergence of the simulated annealing, in fact: The presence of multiple degenerate states with the same energy allows the system to move more easily between these states and avoid getting trapped in less favorable states, and it leads to a multiplicity of solutions, favoring convergence towards equilibrium states or optimal configurations in interacting quantum systems; and on the other hand, it allows simultaneous exploration of these states that can help find solutions more rapidly. Degeneracy can also allow the system to better withstand temperature fluctuations and other disturbances, thus promoting a more stable convergence to the optimal state. These advantages can lead us to a significant improvement in the performance of simulated annealing when applied to degenerate quantum systems. All this has led us to consider another structure of the simulated annealing algorithm giving rise to a new simulated annealing method with a degenerate quantum system.

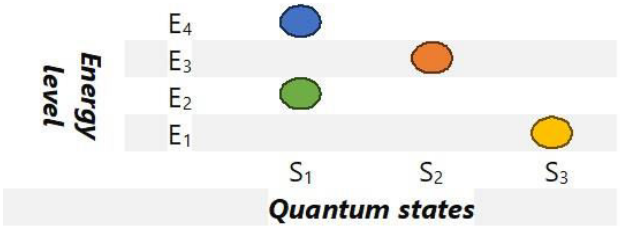

Figure-2. Case of non-degenerate energy levels.

In Figure-2, each energy level (represented by E1, E2, E3, E4) corresponds to a unique quantum state (represented by the circle "O"). There is only one possible configuration for each energy level.

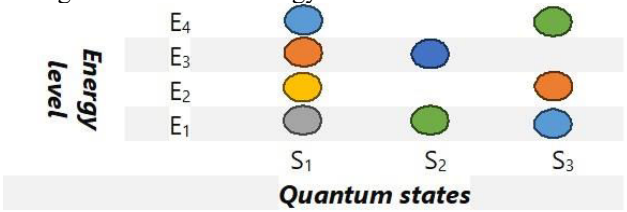

Figure-3. Case of degenerate energy levels.

In Figure-3, several quantum states (represented by the "O" circles) can have the same energy (represented by E1, E2, E3, etc.). There is a multiplicity of possible configurations for each energy level due to degeneracy.

B. The Proposed Simulated Annealing with Degenerate Quantum System

In this context, we propose an innovative method of simulated annealing, which is based on the degeneracy properties of quantum systems. This method, named Degenerate Quantum System Simulated Annealing (SA-DQS), is positioned as a promising extension of standard simulated annealing to solve optimization problems.

The SA-DQS consists of two distinct processing subsystems: the *Non-Degenerate Processing System (NDPS)* and the Degenerate Processing System (DPS).

a) **Non-degenerate processing system (NDPS):** This subsystem operates like a traditional simulated annealing algorithm. It evolves a non-degenerate state using carefully selected and varied neighbor functions to search for new neighboring states.

b) **Degenerate processing system (DPS):** When the NDPS identifies degenerate states, the DPS intervenes. It explores alternative states of the same

energy, generating and evaluating several degenerate states to identify the most promising one.

The switch between the *NDPS* and the *DPS* is managed in such a way as to maximize the complementarity of their respective processing. When performance in one system starts to stagnate or diverge from optimal, the process switches to the other system. This complementarity allows for a more robust and dynamic exploration, increasing the chances of convergence towards an optimal global solution.

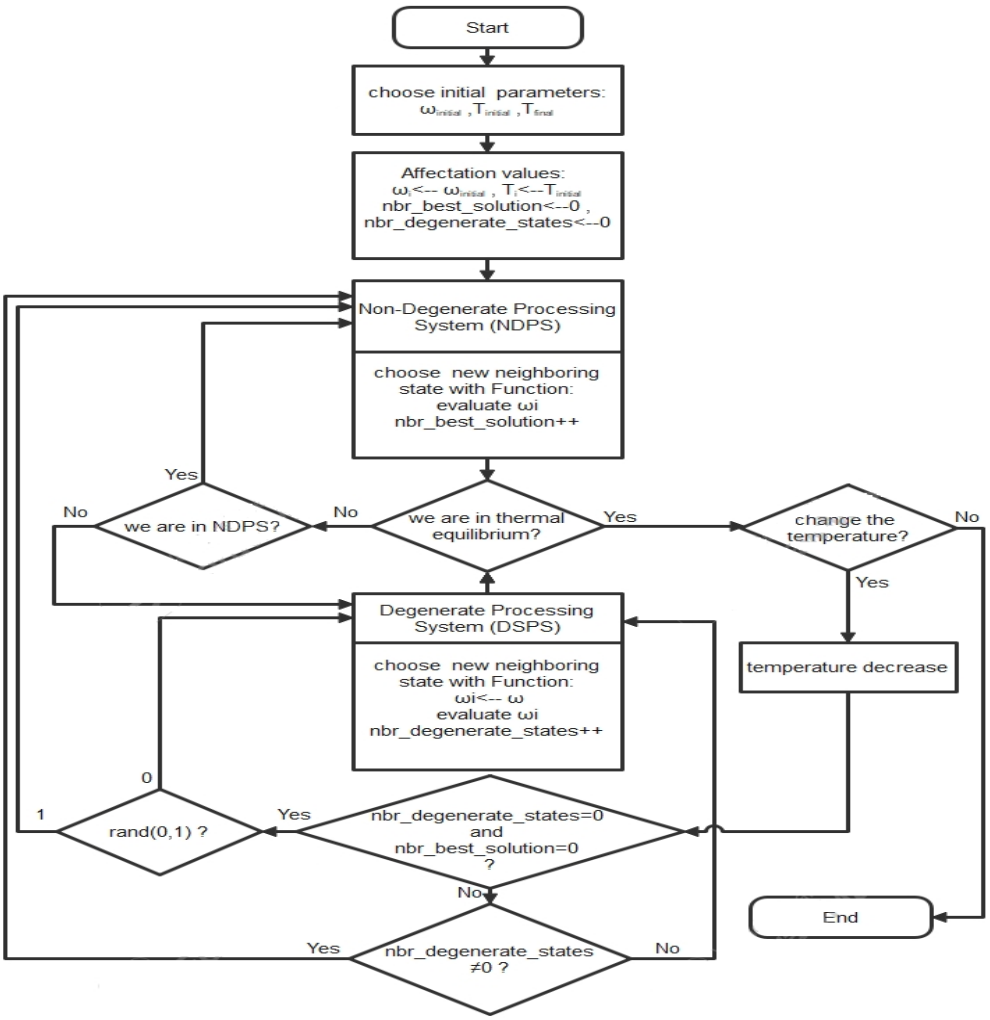

Figure-4. The flowchart of the SA-DQS algorithm.

By integrating these two subsystems, the SA-DQS maintains a dynamic balance between exploration (via the *DPS*) and exploitation particularly suitable for complex optimization problems where degenerate solutions play a crucial role. More specifically, we apply SA-DQS to the TSP. To illustrate how the Degenerate Quantum Simulated Annealing (SA-DQS) algorithm can be applied to the TSP, we will establish an analogy between the concepts of the physical system (SA-DQS) and those of the optimization problem (TSP) as follows: an exploration (via *NDPS*), while ensuring stable and efficient convergence. This method is

Table-1. The analogy between the concepts of the physical system (SA-DQS) and those of the TSP.

SA-DQS (Physical System)	TSP (Optimization Problem)
Non-degenerate state (NDPS)	Unique path covering all cities with a specific total distance
Degenerate state (DSPS)	Different paths covering all cities with the same total distance (energy)
System energy	Total distance of the path
Evolution toward thermal	Exploration of solutions

equilibrium	until convergence or stagnation
Temperature	Control parameter of the global search process (annealing schedule)
Convergence towards the global optimal state	Convergence towards the shortest path covering all cities

This analogy highlights the similarities between a physical system, such as a quantum system with degenerate energy levels, and an optimization problem like the TSP. It shows how physical concepts can be applied to solving optimization problems and how the new SA-DQS method can be adapted to tackle these problems.

The flowchart of Figure-4 describes the steps of the algorithm (SA-DQS), starting with the initialization of parameters such as initial state and initial and final temperatures. Then it moves on to the Non-Degenerate Processing System (NDPS) where a new neighboring state is chosen and evaluated and then accepted according to the Metropolis rules. If the system reaches thermal equilibrium, the temperature is adjusted and, if necessary, the process switches to the Degenerate Processing System (DPS). In DPS, neighboring states of the same energy are generated and evaluated. The criteria for switching between the two systems are based on the performance and stagnation of the solutions. The process continues until the temperature reaches the final temperature threshold, giving a global optimal state.

In the next sections, we will employ the concepts of the TSP optimization problem corresponding to the physical concepts of SA-DQS.

C. Generation Neighbor Functions

In the SA-DQS algorithm, the Non-Degenerate Processing (NDPS) subsystem plays a crucial role in exploring possible optimal solutions via the generation of new neighboring states. Effective design of these neighbors is essential to improve local research. To maximize the efficiency of this exploration, we have implemented three types of neighbor generation functions, named F1: Transpose cities, F2: Swap cities, and F3: Shift cities. The combined use of these functions is more advantageous than strategies based on a single function, as it allows for a more robust and diversified local search. In addition, each type of function offers a unique result, tailored to different aspects of the problem to be solved. Thus, the integration of the three functions into the NDPS optimizes the ability of the SA-DQS algorithm to avoid local minima and converge towards an optimal global solution.

F1: The Transpose City Pairs feature allows you to rearrange a portion of the path to create a new, potentially better solution by reversing cities between two given indices.

F2: City Pair swap, its principle is to exchange the positions of two specific cities in the path.

F3: Shit P times function moves the elements of a path in a circular fashion, taking the first P elements and putting them at the end, and shifting the other elements to the beginning of the path.

Initial path:

C0	C1	C2	C3	C4	C5	C6	C7

City number: 0 1 2 3 4 5 6 7
City1 = 2 (V2), City2 = 6 (V6)

- Before transposition:

Path:

C0	C1	C2	C3	C4	C5	C6	C7

- After transposition:

Path:

C0	C1	C6	C5	C4	C3	C2	C7

Example1: Case F1_Transpose_Cities

Before swap:

path:

C0	C1	C2	C3	C4	C5	C6	C7

- After swap:

Path:

C0	C1	C6	C3	C4	C5	C2	C7

Example2: Case F2_Swap_Cities

- Before shift

Path:

C0	C1	C2	C3	C4	C5	C6	C7

- After shifting 3 positions:

Path:

C3	C4	C5	C6	C7	C0	C1	C2

Example 3: Case F3_Shift_Cities

In the sub-system DPS of our SA-DQS, we use three specific neighbor generation functions: F1: Transpose cities, F3: Shift cities, and F4: Uniformity distances. To identify the most promising path, we use the neighbor generation function F4: Uniformity Distance. This function respects the balanced distribution of distances between city pairs in a given path. High uniformity indicates that distances between cities are evenly distributed, which can lead to more balanced and efficient routes.

Path 1:

C0 ---- 25 ----> C1 ---- 25 ----> C2 ---- 5 ----> C3 ---- 30 ----> C4 ---- 15 ----> C5 ---- 20 ----> C0
Total distance=120

Path 2:
C0 ---- 20 ----> C1 ---- 21 ----> C2 ---- 9 ----> C3 ---- 20 ----> C4 ---- 20 ----> C5 ---- 30 ----> C0
Total distance= 120
Comparison: Choosing the Optimal Path
Path 2 is more uniform than Path 1➔ Choosing the Path2
Example 4: Case of F4_uniformity_distances

In Figure-5, all neighbors were found to have a total distance around the average of the best total distance, to the total of 300,000 neighbors generated during the temperature level T=0.024 for instance A280. We found 724 neighbors around Emoy (best)=2675, The discovery of 724 neighbors indicates that the DPS is capable of finding degenerate solutions at a low temperature level. Likewise, among the 724 neighbors, a promising neighbor (PN) was chosen according to the F4 function. By evolving PN in the NDPS subsystem, PN reached 3 new neighbors with a better total distance than the previous one, going from 2675 to 2654.

These results show that the SA-DQS algorithm with its DSP and NDPS subsystems works efficiently: the NDPS was able to refine the solution found in the DSP and achieve a significantly improved solution; the transition illustrates the synergy between the two systems, where the DSP provides broad exploration and the NDPS focuses on the intensive exploitation of promising solutions.

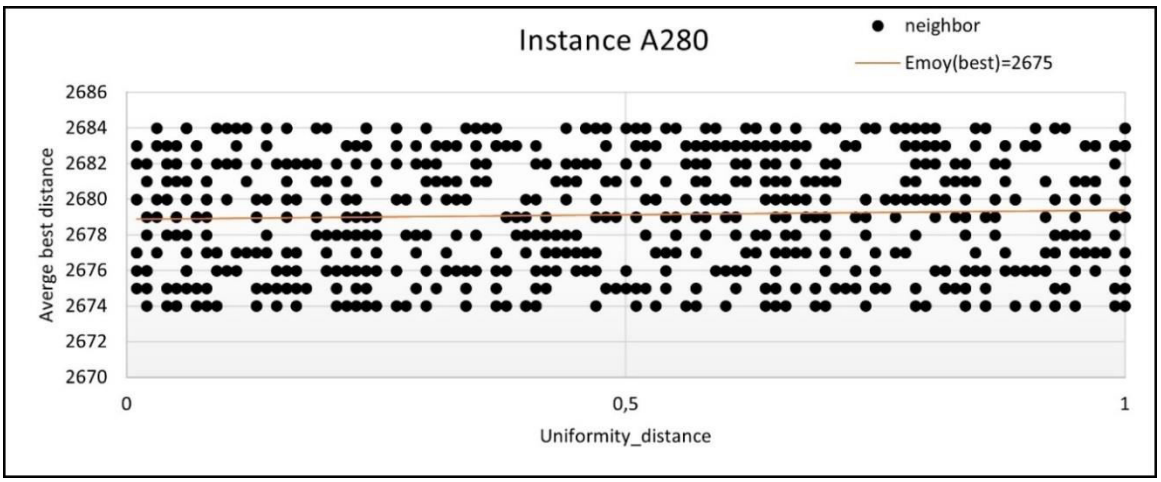

Figure-5. All neighbors found around E_{moy}(best)=2675 during the level T=0.0024 for instance A280.

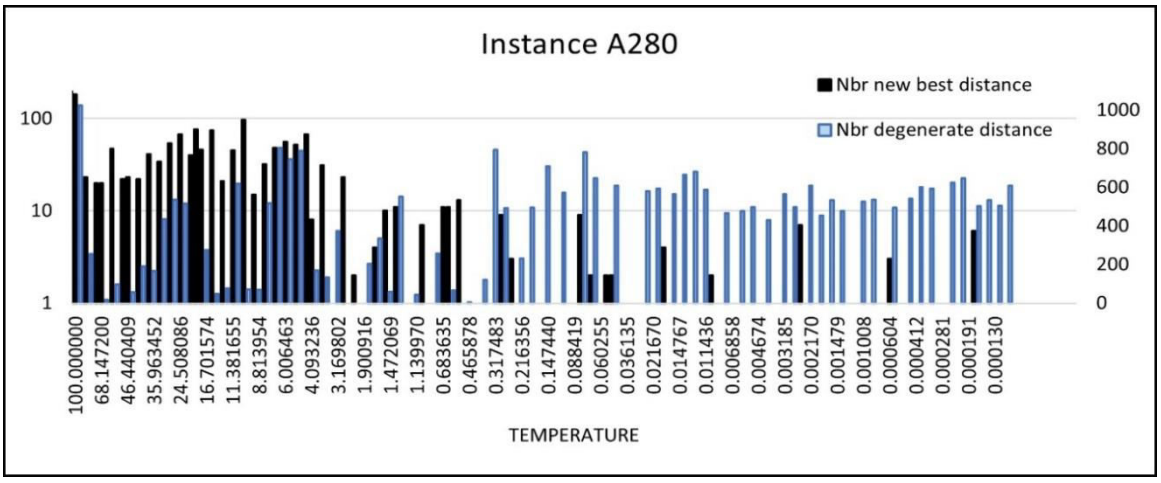

Figure-6. Number of the new best and degenerate distance of SA-DQS for instance A280.

The Figure-6 shows the number (Nbr) of new best and degenerate distances for each level of temperature in the execution of SA-DQS for instance A280 for optimal distance 2589 at 92s. In Temperature Range [100, 4]: An increase in the number of degenerate distances that can reach 1024 similar solutions implies an increase in the number of best distances that can reach 182. In this temperature range, the SA-DQS algorithm can go beyond the local minima and find new best optimal solutions. In Temperature Range [0.216356, 0.088419]: SA-DQS stagnates in terms of discovering new best distances with several degenerate distances varying between 0 and 720.

This may indicate that the algorithm is exploring areas where many similar (degenerate) solutions exist or not but without finding new optimal solutions.

In level 0.077809: SA-DQS goes out of the local minima and finds 9 new best distances.

In the temperature range [0.046662, 0.021670]: SA-DQS is again blocking into local minima, although the DPS continues to find degenerate distances.

In level 0.019069: SA-DQS goes out of the local minima and finds 4 new better distances.

In the level 0.002466: SA-DQS discovers 7 new best distances. This suggests a significant increase in the algorithm's ability to find optimal solutions at this low temperature level.

At the last level 0.000168: SA-DQS does not stagnate and finds 6 new best distances, showing that it continues to discover optimal solutions even towards the final temperature

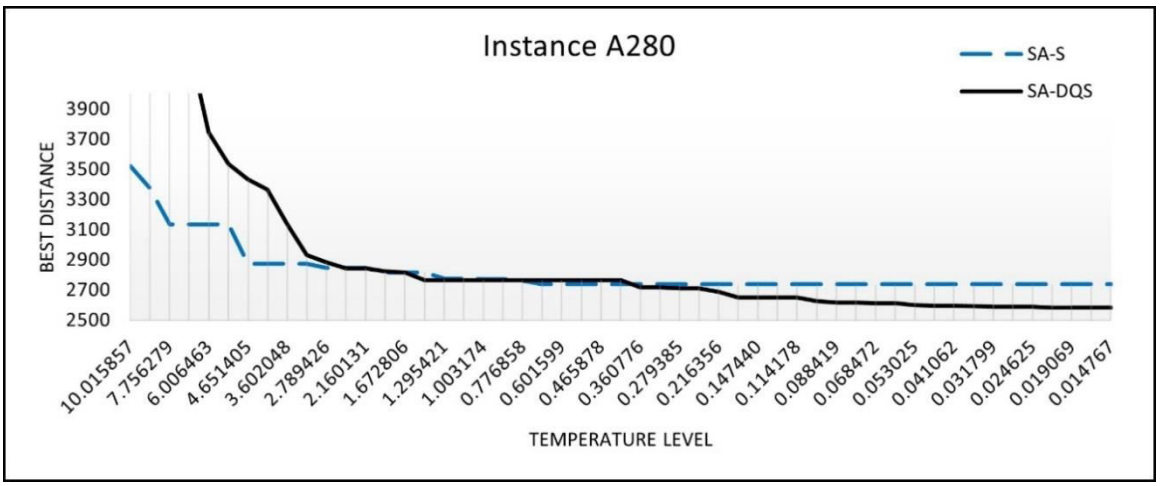

Figure-7. The ability of SA-DQS Vs SA-S to find new best distance solutions between a range temperature [10.015857, 0.014767] for instance A280.

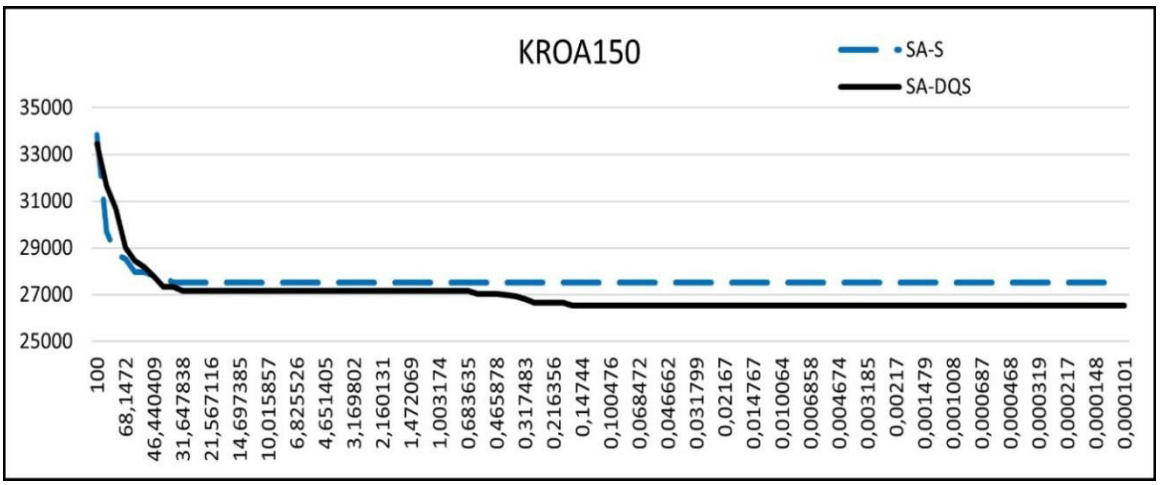

Figure-8. The ability of SA-DQS Vs SA-S to find new best distance solutions between a range temperature [100, 0.0001] for instance KORA150.

In Figure-7 the results obtained for the A280 instance clearly illustrate the dominance of the SA-DQS algorithm over SA-S in the ability to find new best distance solutions for a closed tour of the TSP, between the temperature levels from 10.015857 to 0.014767.

At the beginning of the temperature descent, SA-DQS starts with a distance of 5146, while SA-S starts at 3523. Quickly, SA-DQS reduces the distance significantly, achieving increasingly optimized solutions. For example, when the temperature dropped to around 1.472069, SA-DQS reached a distance of 2767, while SA-S remained stuck at 2818. As the temperature continues to drop, SA-DQS continues to find notable improvements, eventually reaching a distance of 2584 at the 0.021670 level, much better than the final distance of 2737 achieved by SA-S at the 0.683635 level. These results indicate that SA-DQS not only converges more efficiently towards best solutions than SA-S but also that it manages to exit local minima 11

times between the 0.683635 and 0.021670 levels while SA-S remains stuck in the local distance minimum 2737 which represents the final optimal solution of SA-S.In Figure-8, the results of the KORA150 instance between temperature levels 100 and 0.0001, confirm the conclusions observed with the A280 instance. At first, SA-DQS starts with a distance of 33449 and quickly reaches shorter distances, going from 31637 to 26944 to 26822 and finally to a distance of 26524 which is the known best solution of this instance. In comparison, SA-S, starting from an initial distance of 33849, managed to descend to 27511, but remained stuck at this distance until the final temperature, demonstrating its inability to continue improving the solution.

Numerical Simulation Results

To confirm the efficiency and performance of the SA-DQS algorithm, we performed a series of numerical simulations using 9 benchmark instances of the TSPLIB (Reinelt G. 1991)(www.comopt.ifi.uni-heidelberg.de) These instances, ranging in size from 50 to 1200 cities, were subjected to 10 independent trials for each case, and the best result from each round was selected for evaluation. The algorithm, developed in C++, was run on a Windows 11 computer with an "11th generation i5 *CPU=2.4 GHz* and *RAM=8 GB*". As a heuristic method, the performance of SA-DQS depends on several parameters, detailed in Table-2, such as temperature and cooling strategy. The characteristics of the selected instances of the TSPLIB are presented in Table-3.

The SA-S algorithm uses the same parameters of SA-DQS mentioned in Table-2; and the same neighbor functions F1, F2, and F3 and their combination for each instance, except the algorithm which is that of a standard simulated annealing mentioned in Figure-1.

Table-2. Parameters used in the SA-DQS.

Beginning temperature	**100**
Ending temperature	0,0001
Temperature reduction rate	0,88
Number iterations	300000
Number of runs	10

The Figures 9 shows the results of the ability of the SA-DQS and SA-S algorithms to escape local minima and consequently converge to the best optimal distance for a closed tour of the TSP during the optimization process for the instances: BERLIN52, KROA150, A280, D493, RAT575, PCB1173. The number of occurrences of escapees at local minima was counted during each level between the initial and final temperature.

The comparative results between SA-DQS and SA-S on instances less than 100: BERLIN52 demonstrates that SA-DQS is more efficient at escaping local minima: We observe several escapement occurrences at local minima that are particularly high at initial temperatures, with a peak between 45 and 49 occurrences. SA-DQS continues to have notable breakaways, especially with peaks at intermediate temperature steps, and to maintain sporadic breakaways even at very low temperatures. In contrast, SA-S shows a tendency to remain trapped in local minima, with initial breakaways followed by stagnation, even at relatively higher temperatures.

The results on the KROA150 instance between 100 and 200 show a different dynamic between SA-DQS and SA-S in terms of escapes from local minima throughout the cooling process. Although both algorithms start with strong peaks to escape the local minima. After these initial spikes, for SA-DQS, the number of escapes gradually decreases and becomes sporadic, demonstrating that the algorithm is rapidly converging towards high-quality solutions. The number of breakaways for SA-S decreases and becomes more sporadic, with several stages without any breakaways, showing a tendency to stabilize more quickly.

For the A280 instance, SA-DQS shows an interesting behavior with escapes from local minima concentrated in a non-uniform way across temperature levels. Unlike previous instances, it starts with zero escape, indicating an initial phase where it doesn't aggressively explore. However, it shows a rise in power with remarkable peaks that gradually decrease until the final temperature. SA-S, on the other hand, shows a more regular approach and is distributed over the entire temperature range up to the 0.601 level where it stagnates.
In D493 and RAT, 575 instances between 400 and 600 in size, SA-DQS and SA-S show distinct capabilities to escape local minima at different temperature ranges. SA-DQS tends to be more dynamic and uses breakaways intensively at high and medium temperatures. On the other hand, SA-S maintains regular activity and continuous exploration, but with a slightly lower intensity at high temperatures. At low temperatures, both algorithms show a tendency towards convergence, but SA-DQS remains more active in the search for new solutions compared to SA-S which stagnates until the final temperature.

For the PCB1173 instance, SA-DQS and SA-S show distinct capabilities to escape local minima at different temperature ranges. SA-DQS performs better and is more dynamic in the early ranges of high temperatures, while SA-S shows a tendency to stabilize sooner, even if it remains active at high temperatures. At mid and low temperatures, SA-DQS continues to actively explore before gradually stabilizing, while SA-S tends to stabilize more rapidly, showing a rapid decrease in escape activity towards low temperatures; and zero activity from temperature1, 29.

Table-3 shows the performance evaluation of the SA-DQS and SA-S algorithm on the 9 instances of the TSPLIB library, whose optimal solutions are known. SA-DQS shows remarkable performance for several instances of the TSPLIB, reaching or approaching the known optimal values. For small and medium instances such as BERLIN52 and KROA150, the algorithm is extremely

effective. Performance for D493 and RAT575 shows moderate deviations from known optimal and a 5% spacing for the PCB1173 instance. SA-S offers less efficient but faster solutions for small problems, after which its performance degrades noticeably with increasing problem size. SA-DQS outperforms SA-S in terms of accuracy and stability, especially for instances larger than 100 cities, the computation time tends to be faster for both algorithms.

Table-3. Results of our algorithm SA-DQS and SA-S for 9 TSP benchmark instances selected from TSPLIB.

Instance	Length	Optimal	SA-DQS				SA-S			
			Best	Average	Worst	Time(s)	Best	Average	Worst	Time(s)
BERLIN52	52	7542	7542	7542	7542	1	7542	7544	7547	1
ELI76	76	538	538	538,5	540	6,6	544	545,2	546	7,8
BIER127	127	118282	118282	118336,5	118506	7,2	120565	120971,6	121760	4
CH130	130	6110	6112	6124,16	6130	18	6269	6269,8	6286	11
KROA150	150	26524	26524	26580,5	26768	31,4	27250	27573,8	28522	27,2
A280	280	2579	2581	2600,8	2635	55,2	2730	2781,4	2824	186,4
D493	493	35002	35491	35893	36368	245,7	37140	37610,2	38423	234,4
RAT575	575	6773	6939	6975,6	7029	232,5	7352	7422,8	7493	223,6
PCB1173	1173	56892	59671	59976	60335	503,3	66002	64813,4	64777	207,6

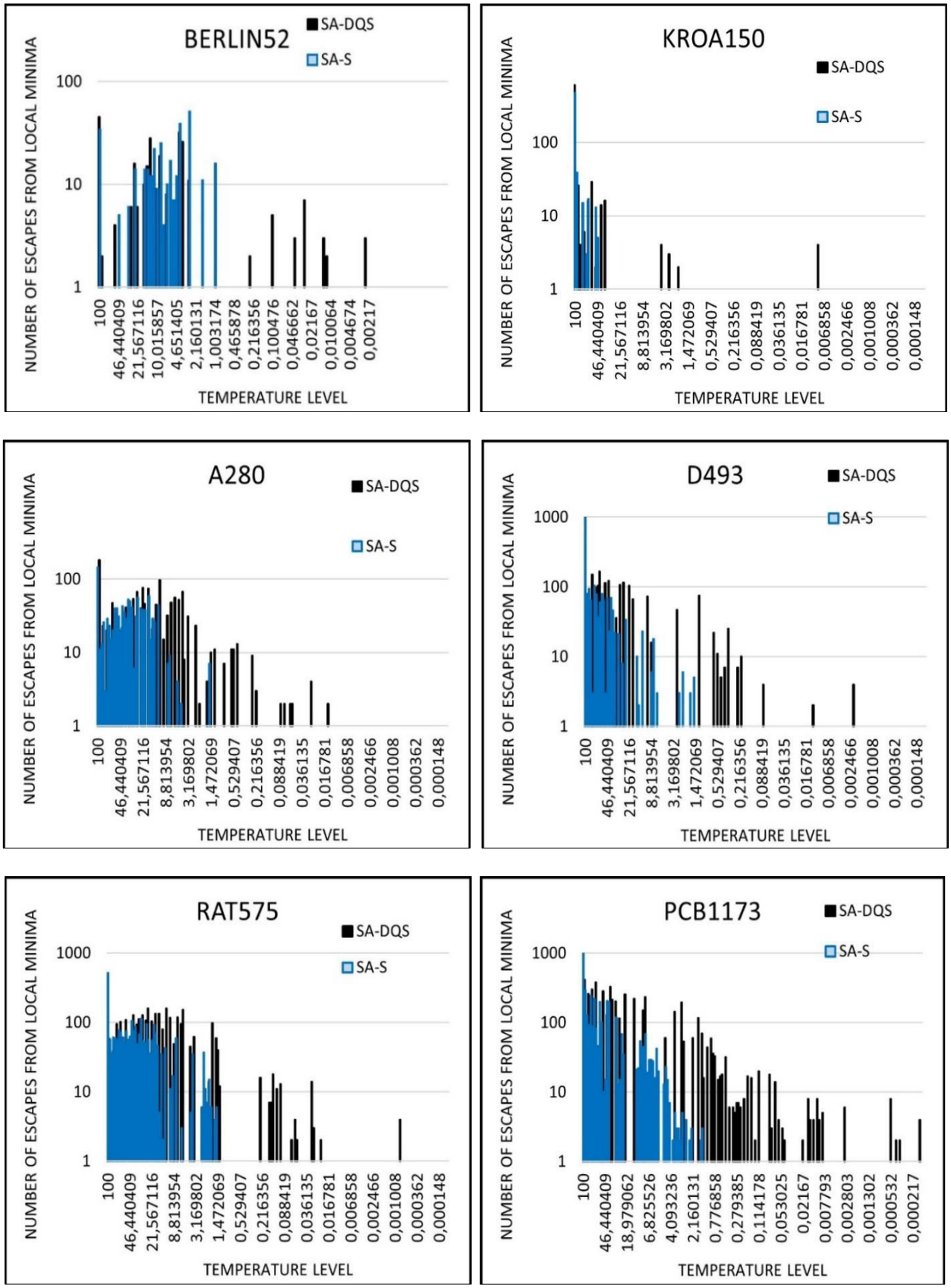

Figure-9. Local Minima Escape Occurrences per Temperature level for 9 Instances: BERLIN52, KROA150, A280, D493, RAT575 and PCB1173.

CONCLUSIONS

The integration of concepts from degenerate quantum systems into optimization algorithms has given rise to an innovative method, SA-DQS, which represents a promising advance for combinatorial problems, in particular TSP. Through a detailed and comparative analysis, we have examined the performance of SA-DQS with standard simulated annealing (SA-S), The results obtained show that SA-DQS surpasses SA-S in terms of proximity to the optimum, getting out of the traps of local minima, stability of solutions, and reduced computation time. This method opens up new perspectives to advance the quality and efficiency of the obtained outcomes, especially in areas where traditional heuristic optimization techniques are showing their limits.

REFERENCES

C. H. Papadimitriou. 1978. Euclidean traveling salesman problem is NP-complete, Theoretical Computer Science. 4: 237-244.

V. B. Lobo, B. B. Alengadan, S. Siddiqui, A. Minu and N. Ansari. 2016. Traveling Salesman Problem for a Bidirectional Graph Using Dynamic Programming. 2016 International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE), Ghaziabad, India. pp. 127-132.

R. Radharamanan, L. I. Choi. 1986. A branch and bound algorithm for the traveling salesman and the transportation routing problems Computers and Industrial Engineering. 11(1-4June): 236-240 https://doi.org/10.1016/0360-8352(86)90085-9

J. Yang, C. Wu, H. P. Lee, Y. Liang. 2008. Solving traveling salesman problem using generalized chromosome genetic algorithm. Progress in Natural Science. 18: 887-892.

Chun-Wei Tsai, Shih-Pang Tseng, Ming-Chao Chiang, Chu-Sing Yang, Tzung-Pei Hong. 2014. A High-Performance Genetic Algorithm: Using Traveling Salesman Problem as a Case. The Scientific World Journal. 2014(Article ID 178621): 14.

R. Baraglia, J. I. Hidalgo, R. Perego. 2021. A hybrid heuristics for the traveling salesman problem. IEEE Transactions on Evolutionary Computation. 5(6): 613-622.

B. A. S. Emambocus, M. B. Jasser, M. Hamzah, A. Mustapha and A. Amphawan. 2021. An Enhanced Swap Sequence-Based Particle Swarm Optimization Algorithm to Solve TSP. in IEEE Access, 9: 164820-164836, doi: 10.1109/ACCESS.2021.3133493.

Y. Cui, J. Zhong, F. Yang, S. Li and P. Li. 2020. Multi-Subdomain Grouping-Based Particle Swarm Optimization for the Traveling Salesman Problem. in IEEE Access, 8: 227497-227510, doi: 10.1109/ACCESS.2020.3045765.

X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu, Q. X. Wang. 2007. Particle swarm optimization-based algorithms for TSP and generalized TSP, Information Processing Letters. 103: 169-176.

Z. Xing and S. Tu. 2020. A Graph Neural Network Assisted Monte Carlo Tree Search Approach to Traveling Salesman Problem. in IEEE Access. 8: 108418-108428, 2020, doi: 10.1109/ACCESS.2020.3000236.

A. H. Gee and R. W. Prager. 1995. Limitations of neural networks for solving traveling salesman problems. In IEEE Transactions on Neural Networks, 6(1): 280-282, doi: 10.1109/72.363424.

K. J. Man, Zhang Yi. 2012. Application of an Improved Ant Colony Optimization on Generalized Traveling Salesman Problem. Energy Procedia. 17(Part A, 2012): 319-325

H. Zhu, X. You and S. Liu. 2019. Multiple Ant Colony Optimization Based on Pearson Correlation Coefficient. in IEEE Access, 7: 61628-61638, doi: 10.1109/ACCESS.2019.2915673.

Z. Wang, X. Geng, Z. Shao. 2009. An Effective Simulated Annealing Algorithm for Solving the Traveling Salesman Problem. Journal of Computational and Theoretical Nanoscience 6(7): 1680-1686. DOI:10.1166/jctn.2009.1230

Lalaoui, M. *et al*. 2018. Simulated Annealing with Adaptive Neighborhood using Fuzzy Logic Controller." International Conference on Learning and Optimization Algorithms.

S. -H. Zhan, Z. -J. Zhang, L. -J. Wang and Y. -W. Zhong. 2018. List-Based Simulated Annealing Algorithm with Hybrid Greedy Repair and Optimization Operator for 0-1 Knapsack Problem. in IEEE Access, 6: 54447-54458, doi: 10.1109/ACCESS.2018.2872533.

Wang L. *et al*. 2019. Enhanced List-Based Simulated Annealing Algorithm for Large-Scale Traveling Salesman Problem. IEEE Access. 7: 144366-144380.

Shi-hua Z., Juan Lin, Ze-jun Zhang, Yi-wen Zhong. 2016. List-Based Simulated Annealing Algorithm for Traveling Salesman Problem. Computational Intelligence and Neuroscience. 2016(Article ID 1712630): 12.

N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. 1953. Equation of state calculations by fast computing machines. Journal of Chemical Physics. 21: 1987-1091.

S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. 1983. Optimization by simulated annealing. Science. 220: 671-680.

V. Cerny. 1985. A thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimization Theory and Applications. 45: 41-51.

Kardar M. 2007. Statistical Physics of Particles. Cambridge: Cambridge University Press.

Reinelt G. 1991. TSPLIB-A traveling salesman problem library. ORSA J. Comput. 3, 376-384. [CrossRef]

www.comopt.ifi.uniheidelberg.de/software/TSPLIB95/tsp/ (accessed on 21 November 2018) Discrete and Combinatorial Optimization-IWR, Heidelberg.