applied sciences

MDPI

Article

A Parallel Meta-Heuristic Approach to Reduce Vehicle Travel Time in Smart Cities

Hector Rico-Garcia [1], Jose-Luis Sanchez-Romero [1,*], Antonio Jimeno-Morenilla [1] and Hector Migallon-Gomis [2]

[1] Department of Computer Technology, University of Alicante, San Vicente del Raspeig, Alicante 03690, Spain; hector.rico@gmail.com (H.R.-G.); jimeno@dtic.ua.es (A.J.-M.)
[2] Department of Computer Engineering, Miguel Hernandez University, Elche, Alicante 03202, Spain; hmigallon@umh.es
* Correspondence: sanchez@dtic.ua.es

Abstract: The development of the smart city concept and inhabitants' need to reduce travel time, in addition to society's awareness of the importance of reducing fuel consumption and respecting the environment, have led to a new approach to the classic travelling salesman problem (TSP) applied to urban environments. This problem can be formulated as "Given a list of geographic points and the distances between each pair of points, what is the shortest possible route that visits each point and returns to the departure point?". At present, with the development of Internet of Things (IoT) devices and increased capabilities of sensors, a large amount of data and measurements are available, allowing researchers to model accurately the routes to choose. In this work, the aim is to provide a solution to the TSP in smart city environments using a modified version of the metaheuristic optimization algorithm Teacher Learner Based Optimization (TLBO). In addition, to improve performance, the solution is implemented by means of a parallel graphics processing unit (GPU) architecture, specifically a Compute Unified Device Architecture (CUDA) implementation.

Keywords: smart cities; meta-heuristics; travelling salesman problem; TLBO; parallelism; GPU

Citation: Rico-Garcia, H.; Sanchez-Romero, J.-L.; Jimeno-Morenilla, A.; Migallon-Gomis, H. A Parallel Meta-Heuristic Approach to Reduce Vehicle Travel Time in Smart Cities. *Appl. Sci.* **2021**, *11*, 818. https://doi.org/10.3390/app11020818

Received: 12 November 2020
Accepted: 13 January 2021
Published: 16 January 2021

1. Introduction

The Smart City concept involves providing the urban environment with smart infrastructure, technology, and procedures to ameliorate the quality of life of inhabitants from an integrated perspective. Among the different aspects that have an impact on improving the quality of life, several action fronts can be noted related to the movement of people and goods, that is, traffic in the urban or metropolitan environment [1]. Indeed, effective traffic management results in a reduction in the travel times of citizens in their vehicles, which in turn has a positive impact on reducing the stress levels of drivers and optimizing the arrival at their workplaces for the effective performance of different professional activities. In addition, effective traffic management also entails better use of existing infrastructure and a reduction in pollution levels, an issue that is becoming increasingly relevant given people's awareness of environmental care and the impact on health.

The problem of traffic in cities becomes more relevant if we take into account the displacements that must be made by carriers and couriers to take different goods from headquarters to different delivery points, whether they are supermarkets, offices, private or homes. It should also be pointed out that the problem will affect the different platforms of autonomous vehicle fleets in the near future, in which the driver takes a passive role and the vehicle makes the decisions regarding the route to be followed to reach its destination in the shortest time.

The large amount of data and measurements that Internet of Things (IoT) devices can provide allows researchers to accurately model the different routes within an urban environment [2]. In this way, as explained in [3,4], smart cities should provide user-centered

mobility services, implementing intelligent and/or automated transportation systems that incorporate strategies of artificial intelligence and technologies, such as the IoT and Physical Internet (PI). The integration of different technologies, such as RFID and LoRaWAN (Long-Range WAN), to implement PI oriented to provide Mobility-as-a-Service is demonstrated in [5].

With the information available, routes are not only determined by the distance between different geographical points, but can also incorporate aspects related to the expected time of arrival, which include, in addition to the distance itself, traffic lights and their crossing times, pedestrian crossings, and possible traffic jams.

A correspondence can be established between this problem and the classic combinatorial travelling salesman problem (TSP), in this case applied to car movements within urban environments. This problem can be informally described in the following manner: "Given a list of geographical points and the distances between each pair of points, what is the shortest possible path that passes one and only one point and returns to the starting point?".

The problem has given rise to a wide variety of research, and various algorithms and heuristics have been developed to try to reduce the complexity when obtaining solutions. In this paper, the aim is to provide an optimal or suboptimal solution to TSP instances applied to smart city environments, using a modified version of the metaheuristic optimization algorithm Teacher Learner Based Optimization (TLBO). In addition, performance is improved by implementing the solution on a parallel graphics processing unit (GPU) architecture, specifically a Compute Unified Device Architecture (CUDA) implementation.

This paper is organized as follows: Section 2 provides a review of state-of-the-art traffic congestion management in smart cities. Section 3 introduces a formal description of the TSP and its similarity with the problem of managing traffic in urban environments. Section 4 describes the TLBO method applied to both continuous and discrete domains. Section 5 shows the manner in which TLBO is implemented on a parallel CUDA architecture to enhance its performance. Section 6 depicts the results of the parallel TLBO when applied to a set of real scenarios from a well-known benchmark. Finally, in Section 7 conclusions of the research work are outlined and future lines of research are proposed.

2. The Concern of Traffic Management in Smart Cities and Urban Environments

One of the main problems to be addressed in an urban environment is vehicle traffic management. Although freight traffic is almost inevitably composed of trucks and vans, passenger movement throughout a city is still mostly operated by private vehicles [6]. As an example, 45% of American people had no access to public transportation in 2018 [7].

Numerous issues regarding the problems associated with car traffic in urban environments are currently being studied. Because the main objective in designing a smart city is to improve "personal satisfaction by utilizing innovation to enhance the proficiency of administrations and address occupants issues" [2], the movements of inhabitants through the city must be managed in such a way that travel time is reduced according to the specific needs of each inhabitant, but concerns about pollution and well-being must also be addressed. The same issues apply to freight transport, because transporters must make their deliveries in an efficient way, trying to reduce delivery times, pollution rates, and stress and tiredness of drivers.

Several research works can be found that deal with the possibility of monitoring traffic in urban environments by means of different technologies and methods. In the work of Rizwan et al. [2], an inexpensive real-time traffic management system is proposed to provide a smart service by activating traffic indicators to instantly update the traffic information. Inexpensive vehicle detection sensors are embedded in the middle of the road every 500 m or 1000 m; IoT is used to quickly acquire traffic information and send it for processing; and real-time flow data is sent for Big Data analysis. In the work of Fernandez-Ares [8], an approach was developed which tracks the movement of people and vehicles by monitoring the radioelectric space, capturing Wi-Fi and Bluetooth signals supplied by smartphones or on-board hands-free devices. In the work of Sendra et al. [9], a collaborative

Long-Range (LoRa)-based sensor network to monitor the pollution levels in smart cities is developed. The system consists of geo-located nodes embedded in vehicles and fixed places to monitor temperature, relative humidity, and concentration of CO_2 in urban environments. The system uses the gathered data to generate the necessary orders to traffic signs and panels that control the traffic circulation. In Kazmi et al. [10], a methodology is proposed that uses VITAL-OS, an IoT platform that enables collection, integration, and management of data streams coming from multifunctional IoT devices and data sources. The information gathered is analyzed to detect traffic noise events; if the noise level on a particular road is higher than a certain threshold, the required traffic signal junction is managed for traffic re-routing to alternative paths. In Kök et al. [11], a deep learning model is proposed for analyzing IoT smart city data. The model is based on Long Short Term Memory (LSTM) networks to predict future values of air quality. In Jabbarpour et al. [12], an IoT application called Intelligent Guardrails is shown. It uses vehicular networks to identify traffic situation of the roads, and incorporates electronic and mechanical techniques to increase the number of lanes of the congested side of a highway while decreasing the lanes on the non-congested side. In Singh et al. [13], a visual Big Data analytics framework for automatic detection of bikers who do not wear a helmet in city traffic is proposed. The paper discusses the issues involved in visual Big Data analytics for traffic control in a city on a surveillance data scope. In Pawłowicz et al. [14], an approach of a traffic management system is proposed. It includes 5G communication, RFID-based parking space monitoring, and cloud services for supervision and machine learning. In Rathore et al. [15], a system for a smart digital city that uses IoT devices for collecting city data and Big Data analytics for acquiring knowledge is proposed. A model is proposed to develop a system that can handle a large volume of city data and provide guidelines to the local authorities. The system implementation involves several stages, including data generation and gathering, aggregation, filtering, classification, preprocessing, computing, and decision making. The gathered data from smart components within the city is processed in real-time to achieve a smart service using Hadoop under Apache Spark. Data generated by smart homes, smart parking areas, weather systems, pollution monitoring sensors, and vehicle networks are used for analysis and testing. In the work of Behnke and Kirschstein [16], a study on the effect of path selection in emission-oriented urban vehicle routing is presented, which is composed of a heterogeneous road network with regard to speed and acceleration frequency. An algorithm to determine all means of emission minimization for pairs of origin and destination nodes given a specific vehicle is proposed and tested using the road network from the city of Berlin.

In Ehmke et al. [17], two data-driven approaches for determining time-dependent emission minimizing paths in urban environments are proposed. Their performances are compared with respect to computing efficiency and solution quality. On average, emissions-minimizing methods can reduce emissions by approximately 3.5% compared to distance-minimizing paths, and 5.0% with regard to minimum time-dependent paths. The emissions-optimized path is roughly 4% longer than the paths generated by distance-based methods, and around 6% longer with regard to travel time compared to travel-time optimized paths.

In the work of Suzuki [18], attention is paid on the so-called pollution routing problem (PRP), which tries to minimize the fuel consumption or pollutants emission of trucks. The final goal is to develop a decision support system of pollution vehicle routing for eco-friendly enterprises. The work identifies, by analyzing the state-of-the-art PRP and gathering expert opinions from carrier managers, a practical PRP model that uses a minimal subset of the main factors affecting fuel consumption, and then develops a solution for this model. In Ehmke et al. [19], research is focused on the problem of minimizing CO_2 emissions in the routing of a fleet of capacitated vehicles in urban environments. A local search procedure, named the tabu search heuristic, is adapted to solve the problem. The research uses instances from a real road network dataset and 230 million speed observations. In Kramer et al. [20], a metaheuristic approach, called ILS-SOA-SP, is proposed to solve

the PRP; this method integrates iterated local search (ILS) with a set partitioning (SP) procedure and a speed optimization algorithm (SOA). The proposed method has the benefit of performing combined route and speed optimization within several local search and integer programming components.

As a summary, a wide variety of research has been carried out with regard to the improvement of quality of life in smart cities in terms of traffic management and, consequently, the reduction of fuel consumption, street noise, and pollution.

3. The Travelling Salesman Problem

The TSP has given rise to a wide variety of research, due to its simplicity of description but its complexity at the time of obtaining a solution [21]. If the problem is formulated using graph theory terminology, it can be defined as a graph $G = (V, A)$, where $V = \{v_1, \ldots, v_n\}$ consists of a set of n vertices (nodes) and $A = \{(v_i, v_j)/v_i, v_j \in V, i \neq j\}$ is a set of edges with an associated non-negative cost (distance) matrix $D = (d_{ij})$. The problem is symmetric if $d_{ij} = d_{ji}$ for any pair of vertices $(v_i, v_j) \in A$, and it is said to be asymmetric (ATSP) otherwise.

If I:$\{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ is a bijective function that determines a reordering of vertices v_i in V, the TSP can be defined as obtaining the minimal result for the following calculation:

$$F_{TSP} = \sum_{j=1}^{n-1} d_{I(j)I(j+1)} \quad (1)$$

It is easy to establish an analogy or correspondence between the TSP and the problem of managing traffic in smart cities. For example, the set of nodes can be related to the different city locations where a van or truck must deliver its cargo from a depot. As previously explained, the meaning of *distance* in the case of urban environments is different from the mere concept of geographical distance, because it can also incorporate traffic lights and their crossing times, pedestrian crossings, speed limits, and even dynamic factors, such as temporary traffic jams, road maintenance works, and accidents. In addition, data collected on fuel consumption and pollution related to different roads can be added.

For a set of n nodes, obtaining the optimal solution by means of exhaustive search is a problem that implies $(n - 1)!$ comparisons. As an example, in the case of ten nodes, the required comparisons between possible solutions would be 9! = 362,880, and 15 nodes would imply comparing 14! = 87,178,291,200 possible solutions to obtain the best solution. The extensive list of research papers related to TSP has produced different methods that try to decrease the aforementioned complexity. Heuristic methods are suitable for managing TSP, and much attention has been paid to these types of methods among researchers on optimization. In [22], a review of swarm intelligence applied to graph search problems is contributed, but the paper focuses almost solely on ant colony optimization (ACO) and bee colony optimization (BCO).

In [23], two different modifications of the artificial bee colony (ABC) method are applied to TSP: a combinatorial algorithm, called CABC, and an improved version of CABC, called qCABC. Experiments are performed on a benchmark of 15 TSP instances. The performance of these two algorithms and eight different genetic algorithm (GA) versions is compared. Moreover, the performance of ABC is also compared with the ant colony system (ACS) and bee colony optimization methods. Results show the suitability of CBAC and qCABC algorithms to be applied to TSP problems.

Most of the works related to solving the TSP by means of metaheuristic optimization methods use the ant colony optimization method [24–28], sometimes in a hybrid version with other methods [29]. This is because the natural behavior of ants in the development of a colony can be easily matched to the problem of finding the shortest path between a series of nodes.

4. The TLBO Optimization Method

4.1. Original TLBO Formulation

Rao [30] developed a series of metaheuristic algorithms that do not require the adjustment of initial parameters to operate. Among these algorithms, TLBO imitates the learning process that takes place in the classroom: from the teacher to the students. It is an iterative process that evolves in stages to optimize a function with multiple variables. This process culminates when the conditions that make the solution valid are met.

For each stage, a random population of possible solutions (called individuals) is created. These values are assigned taking into account the domain of the function f. Once these values are assigned for the population, two stages are carried out, as detailed below.

4.1.1. Teacher Stage

Individual i within the population is represented by vector $X(i)$. Therefore, $X(i,j)$ refers to variable j of individual i. The teacher stage aims to assess the function for each member of the population considering its own parameters, $f(X(i))$. The individual with a value closest to the optimum is selected as the teacher (X_{best}). Then, teacher values are used to bring his disciples closer to him by following Equation (2), which also takes into account the mean values of the population. Thus, each individual will have its new (X_{new}) solution.

$$X_{new}(i,j) = X(i,j) + rand(0,1)(X_{best}(j) - TFactor \cdot X_m(j)) \quad (2)$$

As can be observed in Equation (2), the value of $X(i,j)$ is modified according to the distance that exists between the teacher and the mean of the population (X_m). The factor T is a random integer value that can be valued at 1 or 2, and whose calculation is shown in Equation (3). After changing the values of each individual, the function is evaluated again ($f(X_{new}(i))$). Only in the case that the evaluation offers a more optimal solution than the original one, the new values calculated for the individual i are taken.

$$TFactor = round(1 + rand(0,1)) \quad (3)$$

4.1.2. Learner Stage

At the learner stage, again all individuals are evaluated and compared. This time the comparison takes place with another randomly selected student. In this pair-wise comparison, the student with the optimal solution is called *Best Learner* and the other is called *Worst Learner*. Both students are used to set up a possible new individual X_{new} from the original $X(i)$ according to Equation (4). In case this proposal is more optimal than the original one, the new values calculated will replace the original ones for that individual.

$$X_{new}(i,j) = X(i,j) + rand(0,1) \cdot (BestLearner(j) - WorstLearner(j)) \quad (4)$$

In this algorithm only the number of iterations and the number of individuals in the population have to be fixed; therefore, its main advantage is that there are no parameters that have to be adjusted for the algorithm to converge towards an optimal solution, as happens in other metaheuristic algorithms. In the recent years, there has been an increase in the number of publications that highlight the advantages of TLBO for the resolution of a large number of engineering problems [31–34].

4.2. Discrete TLBO

The problem to be solved in this research, the TSP, is of a discrete nature which comes from the appropriate selection of a combination of sites whose sorting satisfies a criterion. In the case of this research, the aim is to find the shortest route.

This problem does not fit with the characteristics of the original TLBO, whose domain of solutions is established in a continuous range in which the selection of values taken from a set is not possible. For this reason, it is necessary to make a change in the original TLBO algorithm, so that discrete values taken from a set can be selected. This discrete version of

the algorithm (DTLBO) is based on the research proposed in [35] and substantial changes have been made to adapt it to the TSP case study.

In [35], the whole population consists of 100 individuals. Although the paper does not mention it implicitly, it can be guessed that each subpopulation contains four individuals, so 25 subpopulations are supposed to be formed. The division of a population into different subpopulations is a strategy used in several metaheuristic methods to increase the diversity of the population and, therefore, reduce the probability of being trapped in a local optimum.

4.2.1. Representation of Individuals

The representation of individuals in DTLBO changes substantially from the original algorithm. In this version of the algorithm the individuals represent routes through a set of places. Each variable within an individual corresponds to a node or location. As shown in Figure 1, if a set of eight places has to be visited (v_0, v_1 ... v_7), an individual could be represented as eight connected nodes that determine the next order of visit: $v_7 \rightarrow v_0 \rightarrow v_1 \rightarrow v_6 \rightarrow v_4 \rightarrow v_2 \rightarrow v_5 \rightarrow v_3$.

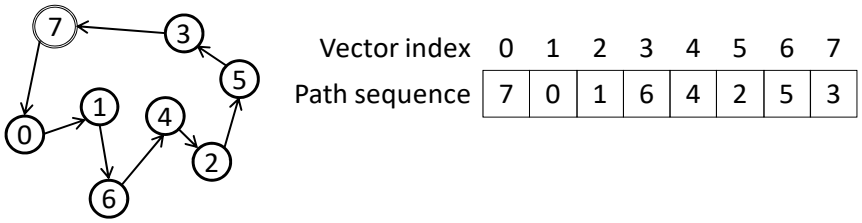

Figure 1. Representation of an individual solution.

In contrast to the work proposed in [35], which uses four completely random subpopulations, in this version of the DTLBO one of the individuals is not generated randomly but is assigned initial values based on a greedy strategy. This is intended to accelerate the convergence of the algorithm and preserve its ability to avoid local minima by maintaining randomness in the subpopulations.

4.2.2. DTLBO Teacher Stage

At this stage, a Partial Teacher is selected for each subpopulation. As with the original TLBO, this teacher is used to improve the students in each subpopulation taking into account the mean value of the subpopulation. In addition, to avoid losing the global vision of the route, a global Teacher is also selected as the best individual of the whole population. It must be taken into account that the calculation of the mean individual could violate the path conditions of the TSP, so a viability condition must be added to its calculation. This condition is performed in the following four steps

Step 1: The number of times a place appears is counted. To store the places that appear more than once, the *TempA* vector is used. Each place is stored in the last position (higher index) that it occupies in the original unfeasible vector; in case a place does not appear or just appears once, the corresponding item remains empty (symbol -). In Figure 2, it can be observed that locations 3 and 5 appear in the last positions they occupied in the original unfeasible vector, that is, positions 5 and 6, respectively.

Step 2: Search for the missing locations. Again, in Figure 2 it can be seen that places 1, 6, and 7 do not appear. The *TempB* vector is used to store these places. In this vector, the index of the place that does not appear is written in the associated item, leaving the remainder of the items empty.

Step 3: A vector *TempC* is created to indicate the places that appear just once. In Figure 2, it can be observed that places 0, 2, and 4 appear in their corresponding positions within *TempC*.

Step 4: Using the above mentioned vectors, a viable individual is constructed by adding to the empty items of *TempA* and *TempC* the places that do not appear in them and that are found in *TempB*.

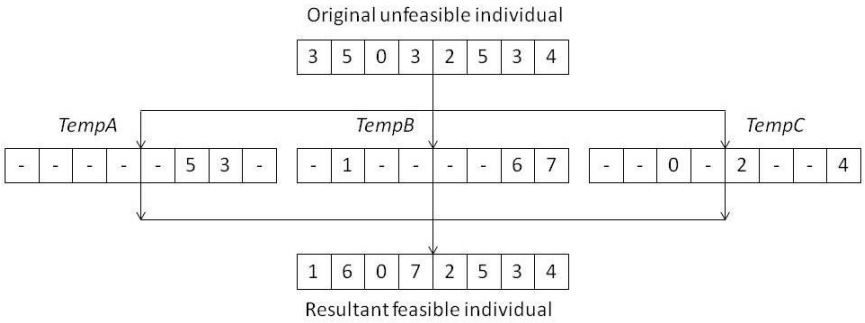

Figure 2. An example of the viability operation.

The crossover operation is used in the DTLBO to generate new individuals from existing ones. This operation is indicated by the symbol $\otimes$. In this version of the algorithm, four different ways of creating new individuals from crossovers were introduced and are shown in Equation (5). In contrast to the work presented in [35], in which a fixed cross is assigned between individuals, this approach makes a random selection for each cross operation. This randomness avoids falling into local minima and increases the variability of populations.

$$\begin{aligned} Xnew(i) &= X(i) \otimes Teacher \\ Xnew(i) &= X(i) \otimes PartialTeacher(i) \\ Xnew(i) &= X(i) \otimes Mean(i) \\ Xnew(i) &= PartialTeacher(i) \otimes Mean(i) \end{aligned} \tag{5}$$

The operation of the crossover can be shown in Figure 3. Suppose that two individuals A and B must generate a new one called A_c. The initial and final positions of the new individual are selected randomly and thus the new individual A_c replaces the selected items by those of individual B. Logically, after this crossover is performed, the viability operation must be applied.

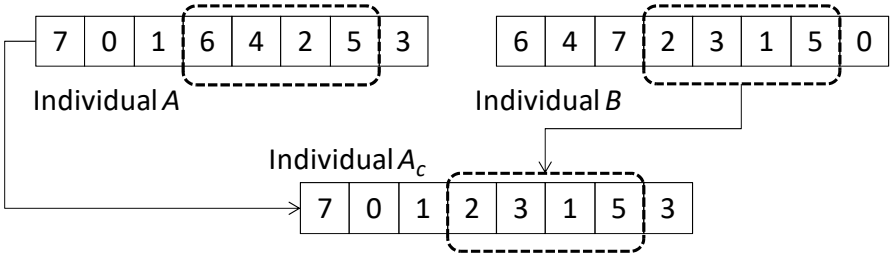

Figure 3. An example of the crossover operation.

To increase the variability of the populations, a mutation operator is included, which is denoted by the symbol Θ (shown in the Equation (6)). Figure 4 shows the functionality of this operator. Given an A_c individual, initial and final positions are randomly selected; in the current example, these are positions 3 and 6. Once these positions are determined, the elements included in this sequence (items of positions 3, 4, 5, and 6) are inverted to create the mutated element A_{cm}.

$$A_{cm}(i) = \Theta A_c(i) \tag{6}$$

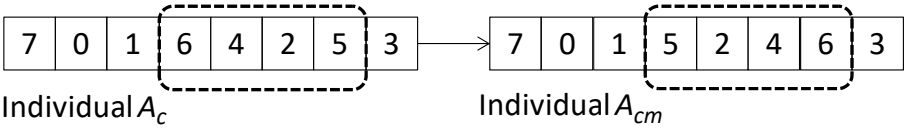

Figure 4. An example of the mutation operation.

4.2.3. DTLBO Learner Stage

Similar to the original TLBO, in the learner stage a random individual k is chosen for each individual i in the subpopulation to be compared with. The new individual is created from Equation (7), where $\otimes$ represents the crossover operation, working in the same way as the crossover operation in the teacher stage. Then it will be necessary to apply the condition of viability on the new individual $X_{new}(i)$ and then apply the mutation operator as described in the teacher stage.

$$Xnew(i) = X(i) \otimes X(k) \quad (7)$$

Although the accuracy of the DTLBO presented in [35] is best when compared with well-known algorithms such as ACO, Particle Swarm Optimization (PSO), and Genetic Algorithm (GA), and only in some specific cases it is slightly beaten by ACO and ABC. It is also the case that its performance is low when cities have a large number of places to visit. For this reason, this investigation focused on a revision of this algorithm based on the improvement of its convergence and on a parallel implementation that allows taking advantage of the performance of the multiprocessing present in the existing GPUs.

5. Parallel TLBO Implementation on GPU

As explained above, applying a series of modifications to the TLBO algorithm can achieve significant results in discrete combinatorial problems, in this case, in the TSP problem. Satisfying results are achieved compared to other algorithms [35].

The changes implemented in the original TLBO cause the different stages to be more complex and add a higher computational cost to the algorithm. As a consequence, the iterations of the DTLBO are significantly more expensive than those of the original TLBO, and therefore the computation time is penalized. With the aim of minimizing the impact of these modifications and the extra cost of computing on the performance of the algorithm, a parallel implementation of the algorithm was developed using a CUDA architecture in a GPU environment. Although parallelization of an algorithm using CUDA is not always the best solution when applying parallelization techniques, the specific features of DTLBO can be exploited to provide a remarkable improvement in performance when compared to sequential solutions using this parallel architecture. Previous research has attempted improvement of metaheuristic methods applied to graph search problems by means of parallel implementations [36].

The initial and fundamental step when proceeding with the algorithm parallelization consists in creating an adequate design of the memory structure and the execution flow to minimize the global thread blockages. If GPU memory is mismanaged, the impact in execution time can be detrimental because transference operations between the different memory levels within the GPU imply a high latency. Therefore, these transferences should be minimized.

5.1. Design of the Memory Organization

The first task that must be carried out to parallelize the algorithm is to conceive an adequate design of the memory structure and the execution flow to minimize the blockage of the different threads executing in parallel within the GPU. Thus, the different GPU memory levels are organized to manage different kinds of data (thread, subpopulation, and global levels) as shown in Figure 5.

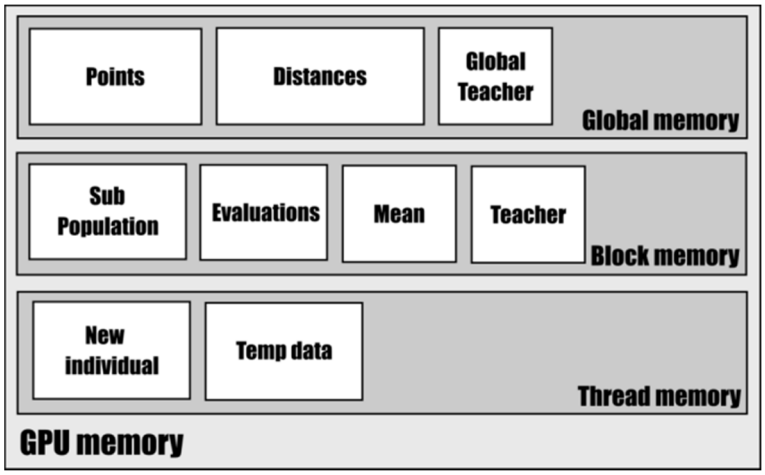

Figure 5. Organization of the Compute Unified Device Architecture (CUDA) memory.

Global memory. This memory stores two main data blocks: an array that contains the required points for generating the individuals of the populations in each block; and an array that contains the whole set of pre-calculated distances between the different points. Global memory also stores the best individual of the whole population and its evaluation (global Teacher), which are globally calculated in each iteration to be used by every individual within the population. Much of the data stored in the global memory will be only read but never modified, and will be used to avoid computing the total distance travelled when evaluating each individual. Only the best evaluation (global Teacher) must be modified if required each iteration.

Block memory. This memory stores the data shared by a subpopulation. The memory is organized in a matrix; each row corresponds to an individual, and each column stores the index of an urban node. An extra column is in charge of storing the individual's solution to avoid repeating the evaluation.

Thread memory. This consists of a private, local memory to each thread which is not shared with the remainder of individuals. The variables stored in this memory are used for the calculations needed at each stage of the DTLBO, and are updated during each of the iterations.

5.2. Execution Flow

An execution flow was devised to minimize the blockages of the threads at the time of synchronization when running the algorithm (*syncthreads*). The TLBO consists of a series of phases linked in such a way that they make it necessary to synchronize the threads to obtain common data for the whole population, such as the mean individual and the best individual (Teacher). In addition, to obtain this information, the reduction technique is utilized to achieve a minimal number of iterations needed to obtain the aforementioned values from the population. To build a parallel execution of the algorithm, each thread corresponds to an individual of the population. The first thread (with index 0) of each subpopulation is responsible for carrying out the operations in the local memory, and the first thread of the first population is responsible for carrying out the necessary operations in the global memory. This is represented in Figure 6. In this manner, conflicts and delays in the access to the different memory levels are avoided.

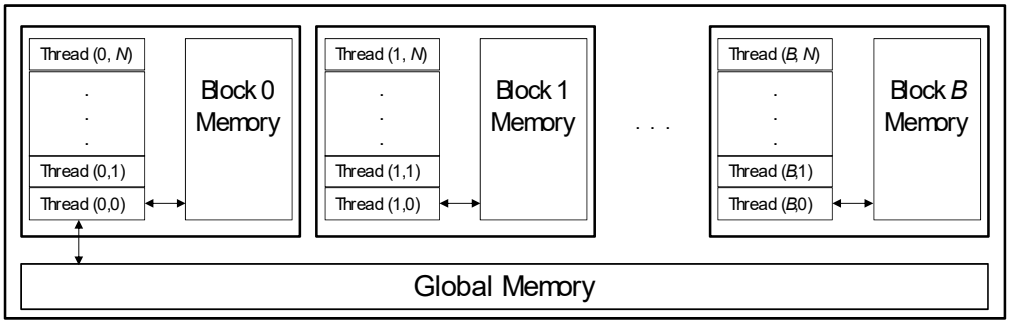

Figure 6. Subpopulation organization and memory accesses.

6. Experimentation

Experimentation was performed to compare the execution time when solving four TSP problems from the TSPLIB library [37]. A sequential implementation and a manycore GPU parallel implementation based on CUDA (9.2 version) were compared. The system used for performing the experiments was equipped with a Pentium i7 3.2 GHz and an NVIDIA GeForce 1060p graphics accelerator, 6 GB GPU RAM, and 32 GB DDR4 RAM. Different scopes were devised for each of the problems by varying the population sizes (64 and 128 individuals) and the number of iterations performed (1000, 5000, and 10,000 iterations). The cities from TSPLIB that were considered for the experiments were Berlin52, Att48, Eil76, and Ch130, with 52, 48, 76, and 130 urban nodes, respectively. They represent real scenarios taken from cities in Europe, USA, and China.

Figures 7–14 depict the results with respect to the CPU and GPU mean time in milliseconds (ms) of 10 blocks of 1000 runs, each with a PC clean reset. Results show that the DTLBO GPU parallel version improves performance up to 6× when dealing with a high number of individuals and iterations. DTLBO reached the optimal path in Att48 (distance = 33,523); in the case of Berlin52, the difference obtained with regard to the optimal was only 2 (7544 versus 7542); in the case of the problems with a higher number of urban nodes, DTBLO obtained 6336 in Ch130 (optimal = 6110), and 552.63 in Eil76 (optimal = 538). The paths obtained for these four real urban scenarios are shown in Figures 15–18.

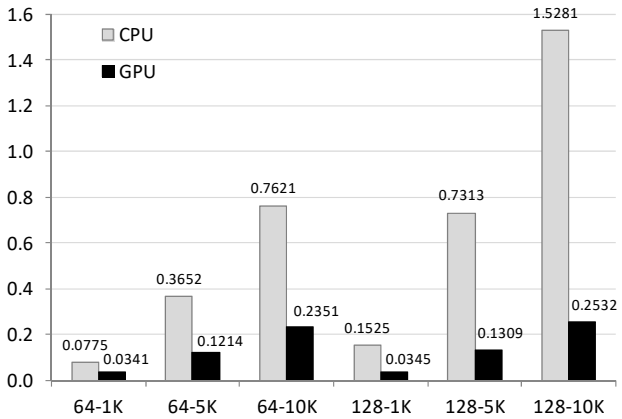

Figure 7. CPU and graphics processing unit (GPU) times (in ms) from Berlin130 with respect to different populations (64 and 128) and iterations (1000, 5000, and 10,000).

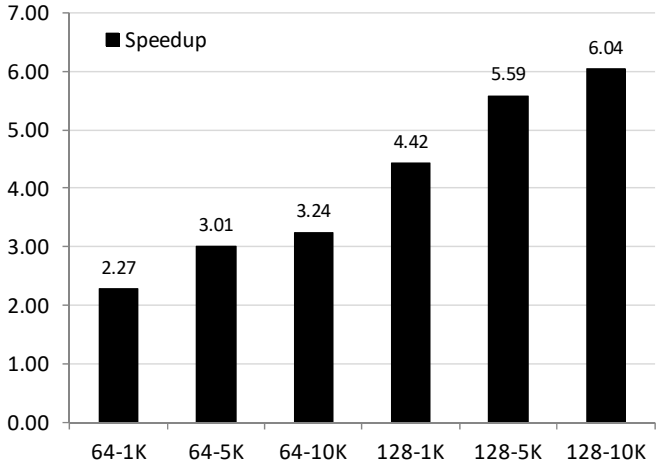

Figure 8. Speedup when comparing CPU and GPU times from Berlin130 with respect to different populations (64 and 128) and iterations (1000, 5000, and 10,000).

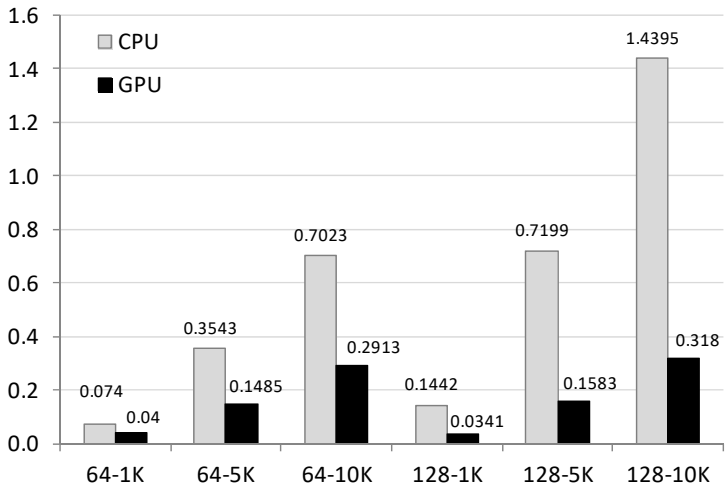

Figure 9. CPU and GPU times (in ms) for Att48 with respect to different populations (64 and 128) and iterations (1000, 5000, and 10,000).

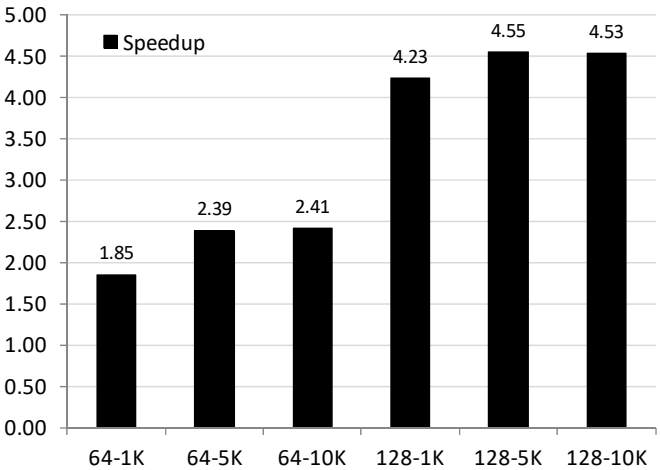

Figure 10. Speedup when comparing CPU and GPU times for Att48 with respect to different populations (64 and 128) and iterations (1000, 5000, and 10,000).

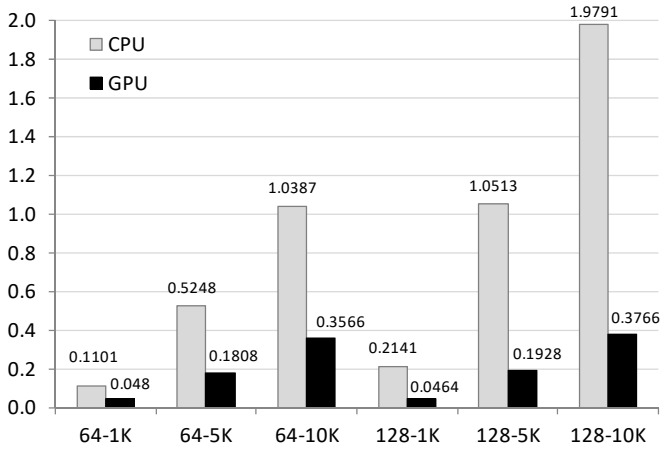

Figure 11. CPU and GPU times for Eil76 (in ms) with respect to different populations (64 and 128) and iterations (1000, 5000, and 10,000).

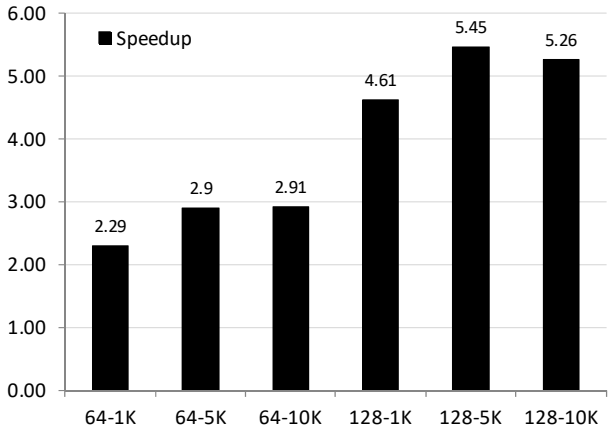

Figure 12. Speedup when comparing CPU and GPU times for Eil76 with respect to different populations (64 and 128) and iterations (1000, 5000, and 10,000).

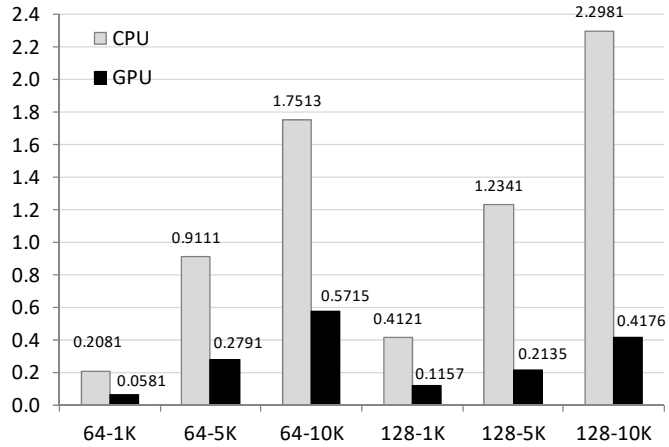

Figure 13. CPU and GPU times (in ms) for Ch130 with respect to different populations (64 and 128) and iterations (1000, 5000, and 10,000).

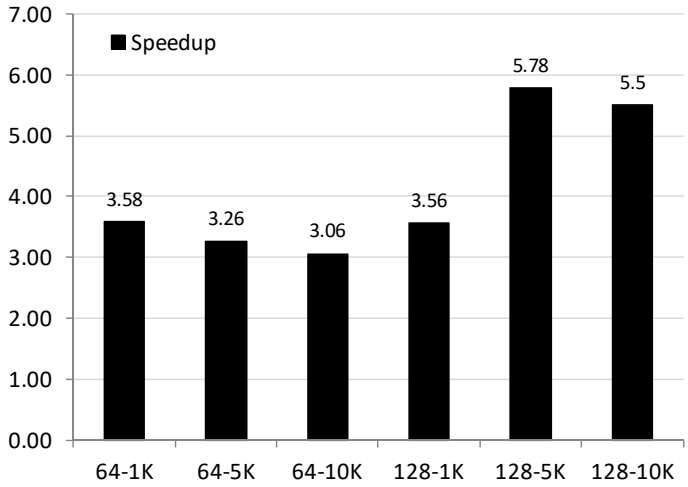

Figure 14. Speedup when comparing CPU and GPU times for Ch130 with respect to different populations (64 and 128) and iterations (1000, 5000, and 10,000).

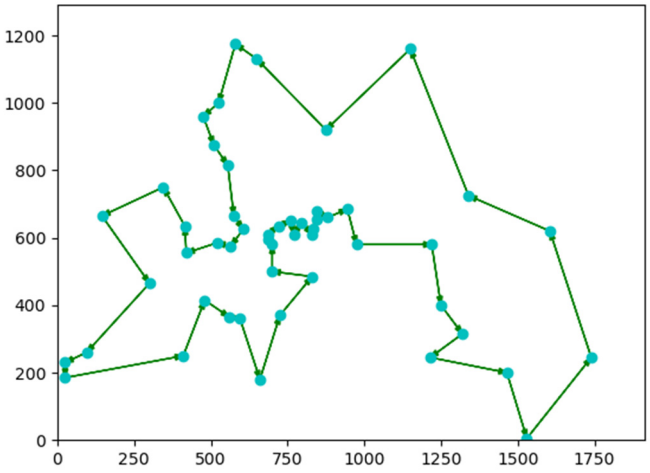

Figure 15. DTLBO solution for Berlin52.

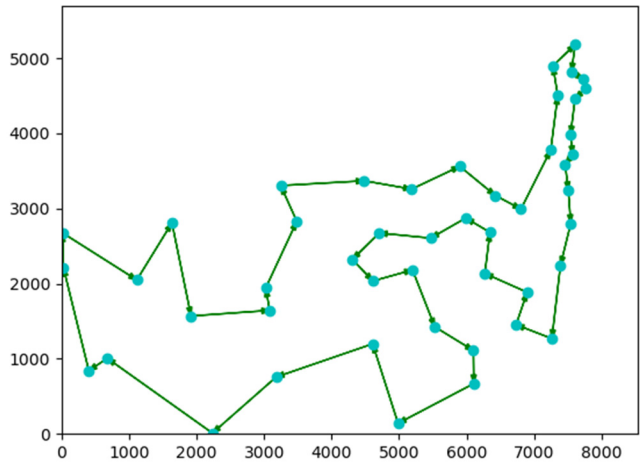

Figure 16. DTLBO solution for Att48.

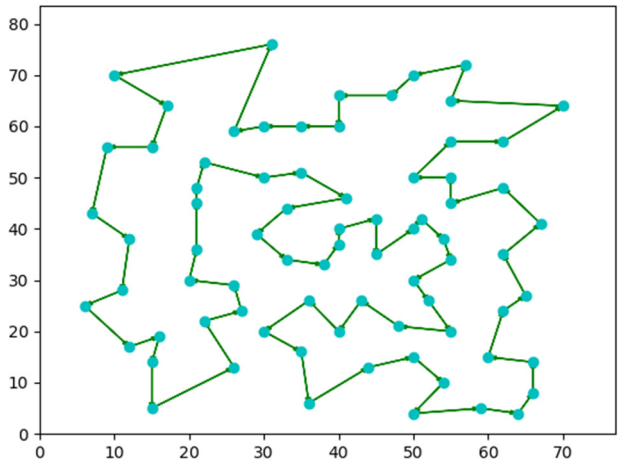

Figure 17. DTLBO solution for Eil76.

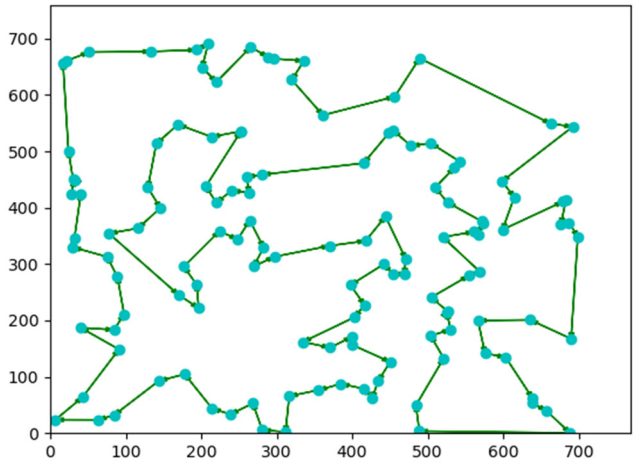

Figure 18. DTLBO solution for Ch130.

The GPU's features allow it to process a high number of threads per block. As a consequence, it was noted that the increase in the population does not have as much time penalty as the CPU, so the GPU times are only affected by the number of iterations. Moreover, the GPU block architecture is highly suitable for subpopulation algorithms, due to the fact that they can be placed in different blocks of the GPU and consequently run in parallel if the number of subpopulations allows it. It is possible for different problems to be executed on the same GPU, achieving a maximal utilization of the GPU resources and a minimal response time. All of the optimal solutions were obtained with populations of 128 individuals. With populations of 64 individuals, it was only possible to generate one of the optimal solutions (the one of the smallest problem).

It was established that population sizes of 64 and 128 are capable of computing all scenarios without memory problems: in the case of 64 individuals, to have a population size close to the number of variables of some problems; and in the case of 128 individuals, to have a population size above almost all of the problems. The maximum of 128 is also related to the limits of memory per block. The data stored at block level to be shared among the whole population for learning, and the comparisons between individuals, limit the population size. Although there are no thread-level memory conflicts, the information needed to make certain calculations, especially for the calculation of valid solutions and the choice of learners, makes it possible for memory conflicts to arise in some of the scenarios.

7. Conclusions

In this research, a parallel implementation of the discrete Teaching Learning Based Optimization algorithm (DTLBO) using a manycore GPU environment to improve the algorithm performance was proposed and applied to provide optimal or suboptimal solutions to the traveling salesman problem. Previous research has found this algorithm to be an excellent option for solving the TSP, but its biggest drawback is its complexity and computational cost when compared to other metaheuristic optimization algorithms. The parallel implementation using a manycore GPU proposed and developed in this research work substantially improved the algorithm performance, achieving important speedups with respect to a sequential implementation. When a high number of individuals and iterations are considered, the speedup is high as $6\times$. The results show that the algorithm is adequate to be applied to problems in which the execution time is one of the determining factors, especially when there are numerous urban nodes in the route that must be traveled in an optimal way.

To summarize, the original DTLBO method was modified by including a greedy strategy when searching for a first initial best individual instead of a purely random generation. When creating new individuals from crossovers, four different ways were considered, with one of them randomly selected each time. Indeed, because the method is time-consuming, another significant contribution of the work consists in accelerating the algorithm by means of a parallel implementation supported by CUDA architecture. The parallelization of this kind of algorithm is not a trivial task and, in fact, the results obtained when parallelizing an algorithm are not always better than those obtained with a sequential implementation. The parallel implementation of DTLBO proposed in this research work achieves satisfactory speedup results.

The developed implementation could be further enhanced in several parts by focusing on improvements in thread block management to optimize the use of GPU resources.

Moreover, although in this research work parallel implementations of the algorithm on CUDA were performed on a desktop computer, these algorithms can be executed in embedded systems inside cars, because some cars already exist with NVIDIA chipsets and support for CUDA, that are used for image recognition within smart driving. With these systems, smart driving could be improved by performing route enhancement using checkpoints within the GPU processing without penalizing the CPU. This is an important issue due to the fact that, in embedded systems in cars, the CPU is a highly demanded resource. Thus, it can be troublesome to run intensive processes on the CPU because the CPU must manage other subsystems in the vehicle. Another line of future research is the adoption of a multi-objective strategy aimed at improving travel time, which would include factors other than distance, such as traffic lights, car accidents, and road works.

Author Contributions: H.R.-G.: Conceptualization, Methodology, Software, Validation, Investigation, writing—original draft, Writing—review & editing. J.-L.S.-R.: Conceptualization, Methodology, Software, Validation, Investigation, writing—original draft, Writing—review & editing, Supervision, Project administration. A.J.-M.: Validation, Investigation, writing—original draft, Writing—review & editing, Supervision, Project administration, Funding acquisition. H.M.-G.: Validation, Investigation, writing—original draft, Writing—review & editing, Supervision, Funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Spanish Ministry of Science, Innovation and Universities and the Research State Agency under Grant RTI2018-098156-B-C54 co-financed by FEDER funds, and by the Spanish Ministry of Economy and Competitiveness under Grant TIN2017-89266-R, co-financed by FEDER funds.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Benevolo, C.; Dameri, R.P.; D'Auria, B. Smart Mobility in Smart City. In *Empowering Organizations, Lecture Notes in Information Systems and Organisation*; Torre, T., Braccini, A., Spinelli, R., Eds.; Springer: Cham, Switzerland, 2016; Volume 11.
2. Rizwan, P.; Suresh, K.; Babu, M.R. Real-Time Smart Traffic Management System for Smart Cities by using Internet of Things and Big Data. In Proceedings of the 2016 International Conference on Emerging Technological Trends (ICETT), New York, NY, USA, 1–6 August 2016.
3. Nikitas, A.; Michalakopoulou, K.; Njoya, E.T.; Karampatzakis, D. Artificial Intelligence, Transport and the Smart City: Definitions and Dimensions of a New Mobility Era. *Sustainability* **2020**, *12*, 2789. [CrossRef]
4. Cruz, C.O.; Sarmento, J.M. 'Mobility as a Service' Platforms: A Critical Path towards Increasing the Sustainability of Transportation Systems. *Sustainability* **2020**, *12*, 6368. [CrossRef]
5. Karampatzakis, D.; Avramidis, G.; Kiratsa, P.; Tseklidis, I.; Oikonomidis, C. A Smart Cargo Bike for the Physical Internet enabled by RFID and LoRaWAN. In Proceedings of the 2019 Panhellenic Conference on Electronics & Telecommunications (PACET), University of Thessaly, Volos, Greece, 8–9 November 2019.
6. Department for Transport. Transport Statistics Great Britain: 2019 Summary. Available online: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/870647/tsgb-2019.pdf (accessed on 8 April 2020).
7. American Public Transportation Association. Public Transportation Facts. Available online: https://www.apta.com/news-publications/public-transportation-facts (accessed on 8 April 2020).
8. Fernández-Ares, A.; Mora, A.; Arenas, M.G.; García-Sanchez, P.; Romero, G.; Rivas, V.; Castillo, P.; Merelo, J. Studying real traffic and mobility scenarios for a Smart City using a new monitoring and tracking system. *Futur. Gener. Comput. Syst.* **2017**, *76*, 163–179. [CrossRef]
9. Sendra, S.; Garcia-Navas, J.L.; Romero-Diaz, P.; Lloret, J. Collaborative LoRa-Based Sensor Network for Pollution Monitoring in Smart Cities. In Proceedings of the 2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC), Rome, Italy, 10–13 June 2019.
10. Kazmi, A.; Tragos, E.; Serrano, M. Underpinning IoT for road traffic noise management in smart cities. In Proceedings of the 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Athens, Greece, 19–23 March 2018.
11. Kök, İ.; Şimşek, M.U.; Özdemir, S. A deep learning model for air quality prediction in smart cities. In Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 11–14 December 2017.
12. Jabbarpour, M.R.; Nabaei, A.; Zarrabi, H. Intelligent Guardrails: An IoT application for vehicle traffic congestion reduction in smart city. In Proceedings of the 2016 IEEE International Conference on Internet of Things (Ithings) and IEEE Green computing and communications (Greencom) and IEEE Cyber, Physical and Social Computing (cpscom) and IEEE Smart Data (smartdata), Chengdu, China, 15–18 December 2016.
13. Singh, D.; Vishnu, C.; Mohan, C.K. Visual Big Data analytics for traffic monitoring in smart city. In Proceedings of the 2016 15th IEEE international conference on machine learning and applications (ICMLA), Anaheim, CA, USA, 18–20 December 2016.
14. Pawłowicz, B.; Salach, M.; Trybus, B. Smart city traffic monitoring system based on 5G cellular network, RFID and machine learning. In Proceedings of the KKIO Software Engineering Conference, Pultusk, Poland, 27–28 September 2018; pp. 151–165.
15. Rathore, M.M.; Paul, A.; Hong, W.-H.; Seo, H.; Awan, I.; Saeed, S. Exploiting IoT and big data analytics: Defining Smart Digital City using real-time urban data. *Sustain. Cities Soc.* **2018**, *40*, 600–610. [CrossRef]
16. Behnke, M.; Kirschstein, T. The impact of path selection on GHG emissions in city logistics. *Transp. Res. Part E Logist. Transp. Rev.* **2017**, *106*, 320–336. [CrossRef]
17. Ehmke, J.F.; Campbell, A.M.; Thomas, B.W. Data-driven approaches for emissions-minimized paths in urban areas. *Comput. Oper. Res.* **2016**, *67*, 34–47. [CrossRef]
18. Suzuki, Y.A. dual-objective metaheuristic approach to solve practical pollution routing problem. *Int. J. Prod. Econ.* **2016**, *176*, 143–153. [CrossRef]
19. Ehmke, J.F.; Campbell, A.M.; Thomas, B.W. Vehicle routing to minimize time-dependent emissions in urban areas. *Eur. J. Oper. Res.* **2016**, *251*, 478–494. [CrossRef]
20. Kramer, R.; Subramanian, A.; Vidal, T.; Cabral, L.D.A.F. A matheuristic approach for the Pollution-Routing Problem. *Eur. J. Oper. Res.* **2015**, *243*, 523–539. [CrossRef]
21. Rego, C.; Gamboa, D.; Glover, F.; Osterman, C. Traveling salesman problem heuristics: Leading methods, implementations and latest advances. *Eur. J. Oper. Res.* **2011**, *211*, 427–441. [CrossRef]
22. Ilie, S.V. Survey on distributed approaches to swarm intelligence for graph search problems. *Ann. Univ. Craiova-Math. Comput. Sci. Ser.* **2014**, *41*, 251–270.
23. Karaboga, D.; Gorkemli, B. Solving Traveling Salesman Problem by Using Combinatorial Artificial Bee Colony Algorithms. *Int. J. Artif. Intell. Tools* **2019**, *28*, 1950004. [CrossRef]
24. Jabir, E.; Panicker, V.V.; Sridharan, R. Design and development of a hybrid ant colony-variable neighbourhood search algorithm for a multi-depot green vehicle routing problem. *Transp. Res. Part D Transp. Envi.* **2017**, *57*, 422–457. [CrossRef]
25. Gan, R.; Guo, Q.; Chang, H.; Yi, Y. Improved ant colony optimization algorithm for the traveling salesman problems. *J. Syst. Eng. Electron.* **2010**, *21*, 329–333. [CrossRef]

26. Shokouhifar, M.; Sabet, S. PMACO: A pheromone-mutation based ant colony optimization for traveling salesman problem. In Proceedings of the 2012 International Symposium on Innovations in Intelligent Systems and Applications, Trabzon, Turkey, 2–4 July 2012.
27. Bai, J.; Yang, G.; Chen, Y.-W.; Hu, L.-S.; Pan, C.-C. A model induced max-min ant colony optimization for asymmetric traveling salesman problem. *Appl. Soft Comput.* **2013**, *13*, 1365–1375. [CrossRef]
28. Głabowski, M.; Musznicki, B.; Nowak, P.; Zwierzykowski, P. Shortest Path Problem Solving Based on Ant Colony Optimization Metaheuristic. *Image Process. Commun.* **2012**, *17*, 7–17. [CrossRef]
29. Mahi, M.; Baykan, Ö.K.; Kodaz, H. A new hybrid method based on Particle Swarm Optimization, Ant Colony Optimization and 3-Opt algorithms for Traveling Salesman Problem. *Appl. Soft Comput.* **2015**, *30*, 484–490. [CrossRef]
30. Rao, R.V.; Savsani, V.J.; Vakharia, D.P. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Comput. Des.* **2011**, *43*, 303–315. [CrossRef]
31. Rao, R.V.; Patel, V. An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems. *Int. J. Ind. Eng. Comput.* **2012**, *3*, 535–560. [CrossRef]
32. Rao, R.V.; Patel, V. Comparative Performance of an elitist Teaching-Learning-Based Optimization algorithm for solving unconstrained optimization problems. *Int. J. Ind. Eng. Comput.* **2013**, *4*, 29–50. [CrossRef]
33. Ebraheem, M.; Jyothsna, T.R. Comparative performance evaluation of Teaching Learning Based Optimization against genetic algorithm on benchmark functions. In Proceedings of the 2015 Power, Communication and Information Technology Conference (PCITC), Bhubaneswar, India, 15–17 October 2015; pp. 327–331.
34. Shah, S.R.; Takmare, S.B. A Review of Methodologies of TLBO Algorithm to Test the Performance of Benchmark Functions. *Program. Device Circuits Syst.* **2017**, *9*, 141–145.
35. Wu, L.; Zoua, F.; Chen, D. Discrete Teaching-Learning-Based Optimization Algorithm for Traveling Salesman Problems. In Proceedings of the MATEC Web of Conferences 128, 02022 EDP Sciences (2017), Zhuhai, China, 23–24 September 2017.
36. Arnautovic, M.; Curic, M.; Dolamic, E.; Nosovic, N. Parallelization of the ant colony optimization for the shortest path problem using OpenMP and CUDA. In Proceedings of the 2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 20–24 May 2013.
37. Universität Heidelberg. Institut für Informatik. TSPLIB. Available online: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ (accessed on 8 April 2020).