 doi:10.20944/preprints202010.0605.v1

Article

A parallel meta-heuristic approach to reduce vehicle travel time in smart cities

Hector Rico-Garcia [1], Jose-Luis Sanchez-Romero [2,*], Antonio Jimeno-Morenilla [3], Hector Migallon-Gomis [4]

[1] Department of Computer Technology, University of Alicante;hector.rico@gmail.com
[1] Department of Computer Technology, University of Alicante; sanchez@dtic.ua.es
[1] Department of Computer Technology, University of Alicante; jimeno@dtic.ua.es
[2] Department of Computer Engineering, Miguel Hernandez University; hmigallon@umh.es
* Correspondence: hector.rico@gmail.com

Abstract: The development of the smart city concept and the inhabitants' need to reduce travel time, as well as society's awareness of the reduction of fuel consumption and respect for the environment, lead to a new approach to the classic problem of the 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?" Nowadays, with the development of IoT devices and the high sensoring capabilities, a large amount of data and measurements are available, allowing researchers to model accurately the routes to choose. In this work, the purpose is to give solution to the TSP in smart city environments using a modified version of the metaheuristic optimization algorithm TLBO (Teacher Learner Based Optimization). In addition, to improve performance, the solution is implemented using a parallel GPU architecture, specifically a CUDA implementation.

Keywords: Smart cities; Meta-heuristics; Travelling Salesman Problem; TLBO; Parallelism; GPU

1. Introduction

The Smart City concept involves providing the urban environment with smart infrastructure, technology and procedures in order to improve the quality of life of citizens from an integrated perspective. Among the different aspects that have an impact on improving the quality of life, it is worth mentioning several action fronts 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 the 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 the different goods from the headquarters to the different delivery points, whether they are supermarkets, offices, private homes, and so on. It should also be taken into account that the problem will affect the different platforms of autonomous vehicle fleets in the near future, in which the driver takes on a passive role and it is the vehicle itself that takes the decisions on the route to be followed to reach its destination in the shortest time.

Today, thanks to the development of IoT and the high sensoring capabilities of different devices, a large amount of data and measurements are available that allow researchers to accurately model the different routes within an urban environment [2].

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 would include, in addition to the distance itself, traffic lights and their crossing times, pedestrian crossings, possible traffic jams, etc.

A correspondence can be established between this problem and the classic combinatorial problem of the Travelling Salesman Problem (TSP), in this case applied to car movements within urban environments. This problem can be formulated informally as follows: "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 great deal of research, and various algorithms and heuristics have been developed to try to reduce the complexity when obtaining solutions. In this work, the purpose is to provide a solution to the TSP applied to smart city environments, using a modified version of the metaheuristic optimization algorithm TLBO (Teacher Learner Based Optimization). In addition, to improve performance, the solution has been implemented using a parallel Graphics Processing Unit (GPU) architecture, specifically a Compute Unified Device Architecture (CUDA) implementation.

This paper is structured as follows: Section 2 provides a state-of-the-art review of the traffic congestion management in smart cities. Section 3 introduces a formulation of the TSP and its analogy with the problem of managing traffic in urban environments. Section 4 describes the TLBO method in both continuous and discrete versions. Section 5 shows the implementation of TLBO on a parallel CUDA architecture so as to improve 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 summarized 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. While freight traffic is almost inevitably operated by trucks and vans, passenger movement throughout a city is still mostly operated by private vehicles [3]. As an example, 45% of American people have no access to public transportation in 2018 [4].

Many issues regarding the problems associated with car traffic in urban environments are currently being studied. Since 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 must apply to the freight transport, as 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 which deal with the possibility of monitoring traffic in urban environments by means of different technologies and methods. In [2], a low cost real-time traffic management system is proposed to provide a smart service by activating traffic indicators to update the traffic details instantly; low cost vehicle detection sensors are embedded in the middle of the road for every 500 meters or 1000 meters; IoT is used to acquire traffic data quickly and send it for processing; real time streaming data is sent for Big Data analytics. In [5], an approach was developed which tracks the movement of people and vehicles monitoring the radioelectric space, capturing WiFi and Bluetooth signals emitted by personal smartphones or on-board hands-free devices. In [6], a collaborative LoRa-based (Long-Range) sensor network to monitor the pollution levels in smart cities is developed; the system is composed by geo-located nodes installed in vehicles and fixed places to monitor temperature, relative humidity and concentration of CO2 in urban environments; by using the collected data, the system generates the required orders to traffic signs and panels which are in charge of controlling the traffic circulation. In [7], a methodology is proposed which uses VITAL-OS, an IoT platform which facilitates collection, integration, and management of data streams stemming from multifunctional IoT devices and data sources; the collected data are analyzed so as to identify traffic noise events; if the noise level on a particular road

exceeds a specified threshold, the required traffic signal junction is controlled for traffic re-routing to alternative paths. In [8], 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 [9], an IoT application named Intelligent Guardrails is presented; it uses vehicular networks to detect traffic condition of the roads and incorporates electronic and mechanical techniques for increasing the number of lanes of the congested side of a highway by decreasing the lanes on the non-congested side. In [10], a framework for visual Big Data analytics for automatic detection of bike-riders without helmet in city traffic is proposed; the paper discusses challenges involved in visual Big Data analytics for traffic control in a city scale surveillance data. In [11], an approach of a traffic management system is proposed, including 5G communication, RFID-based parking space monitoring and cloud services for supervisory control and machine learning. In [12], a system for smart digital city that uses IoT devices for city data harvesting and Big Data analytics for knowledge acquisition is proposed; a smart digital city architecture and implementation model are proposed to implement a system which can handle huge amount of city data and provide guidelines to the local authorities; the system implementation involves several stages including data generation and collection, aggregating, filtration, classification, preprocessing, computing and decision making; the collected data from smart components within the city is processed at real-time to achieve a smart service using Hadoop working under Apache Spark; datasets generated by smart homes, smart parkings, weather systems, pollution monitoring sensors, and vehicle network are used for analysis and testing. In [13], a study on the effect of path selection in emission-oriented urban vehicle routing is presented, which is characterized by a heterogeneous road network with regard to speed and acceleration frequency; an algorithm to determine all emission-minimal paths for an origin-destination pair given a particular vehicle is proposed and tested by using the road network from the city of Berlin.

In [14], two data-driven approaches for determining time-dependent emissions minimizing paths in urban areas are proposed; their performances are compared with regard to computational efficiency and solutions quality; on average, emissions-minimizing methods can reduce emissions by roughly 3.5% over distance-minimizing paths and 5.0% relative to minimum time-dependent paths; the emissions-optimized path is around 4% longer than the paths generated by distance-based methods and around 6% longer in terms of travel time compared to travel-time optimized paths.

In [15], attention is paid on the so-called pollution routing problem (PRP), which tries to minimizes the fuel burn or pollutants emission of trucks; the ultimate goal is to develop a decision support system of pollution vehicle routing for eco-friendly firms; the work identifies, by reviewing the PRP literature and obtaining 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 [16], research is focused on the problem of minimizing expected CO2 emissions in the routing of a fleet capacitated vehicles in urban areas; a local search procedure, named tabu search heuristic is adapted to solve the problem; the research uses instances derived from a real road network dataset and 230 million speed observations.

In [17], a metaheuristic approach, called ILS-SOA-SP, is proposed so as to solve 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.

To summarize, a wide variety of research work are being 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 work, given its simplicity of description but its complexity at the time of obtaining a solution [18]. If the problem is formulated using graph theory terminology, it is defined as a graph $G = (V, A)$, where $V = \{v_1,...,v_n\}$ is a set of n vertices (nodes)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2020 doi:10.20944/preprints202010.0605.v1

and $A = \{(v_i, v_j) / v_i, v_j \in V, i \neq j\}$ is a set of edges that has an associated non-negative cost (distance) matrix $D = (d_{ij})$. The problem is said to be symmetric if $d_{ij} = d_{ji}$ for any pair of nodes $(v_i, v_j) \in A$, and is asymmetric (ATSP) in any other case.

If $I:\{1, ..., n\} \rightarrow \{1, \ldots, n\}$ is a bijective function that defines a reordering of vertices v_i in V, the TSP can be defined as minimizing the following function:

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

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 explained before, the meaning of *distance* in case of urban environments is different from the mere concept of geographical distance, since it can moreover incorporate traffic lights and their crossing times, pedestrian crossings, speed limits, and even dynamic factor such as temporary traffic jams, road maintenance works, wrecks, and so on. In addition, data collected on fuel consumption and pollution related to different roads could be added.

Given a set of n nodes, finding the best solution by exhaustive search would involve a complexity of the order of $(n - 1)!$ That is, in case of ten nodes, it would be necessary comparing 9! = 362,880 possible solutions, while 15 nodes would imply comparing 14! = 87,178,291,200 possible solutions so as to obtain the optimal one. The extensive list of research works related to TSP has produced different methods that reduce this complexity. Heuristic methods are suitable for dealing with TSP, and much attention has been paid to these types of methods among researchers on optimization. In [19], a survey of swarm intelligence applied to graph search problems is contributed, but the paper is almost exclusively focused on Ant Colony Optimization (ACO) and Bee Colony Optimization (BCO).

In [20], two different modifications of the Artificial Bee Colony (ABC) method are proposed for TSP: a combinatorial algorithm, called CABC, and an improved version of CABC, called qCABC; experiments are performed on a set of 15 TSP test problems; the performance of these two algorithms and eight different GA (Genetic Algorithms) versions is compared; moreover, the performance of ABC is also compared with 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 give solution to TSP by means of metaheuristic optimization methods use the Ant Colony Optimization method [21 − 25], sometimes in a hybrid version with other methods [26]. 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 points.

4. The TLBO optimization method

4.1. Original TLBO Formulation

TLBO is a metaheuristic optimization method developed by Rao [27]. It is based on the natural behavior of a *teacher* and a set of students (*learners*) when performing a teaching-learning process by means of an iterative series of stages. The aim of this cyclic behavior is the optimization of a function *f*. A population of individuals is initially created, and each one of them is then assigned a random value for each one of the design variables defined in function *f*.

After this initial assignment, the method iterates through the following two stages until a termination criterion is fulfilled.

4.1.1 Teacher stage

In this stage, each individual *i* evaluates the function *f* with its **ow**n parameters, $f(X(i))$, and the one with the optimal (usually minimal) value is designed as the *Teacher* of the population. Each

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2020 doi:10.20944/preprints202010.0605.v1

individual updates its parameters according to the *Teacher* parameters and the mean population parameters.

Therefore, the *Teacher* (X_{best}) of the current generation is used to create a new version of each individual (X_{new}) according to the following equation:

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

In (2), $X(i, j)$ is the design variable j of individual i. It is modified by using the value of variable j from the *Teacher*, $X_{best}(j)$, the variable mean from the whole population, $X_m(j)$, and the *TFactor*. This last parameter adopts the integer value 1 or 2 according to the following expression:

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

After changing the variables of an individual i, it is submitted for evaluation. In case that evaluation of f ($X_{new}(i)$) provides a better (lower) result than that of the original values, the new values of the design variable replace the old ones for that individual i.

4.1.2 Learner stage

After the teacher stage, each individual is compared with a random competitor from the population. Both individuals are evaluated. The individual with a better evaluation is designed as the *Best Learner*, and the other one is designed as the *Worst Learner*. They are used to generate a new individual by means of the following expression:

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

When every $Xnew(i, j)$ is generated, this new individual is evaluated and compared with the original individual $X(i)$. If the evaluation of the new individual is better than that of the original one, the values of the old one are replaced by the new values in the design variables.

TLBO has an important advantage over other population-based heuristic algorithms: it has no algorithm-specific controlling or tuning parameters. Indeed, only population size and generations (number of iterations) should be configured. In recent years, different research works have shown that TLBO is more efficient than other optimization methods [28 − 31]. Hence, research related to TLBO has reached a remarkable increase in recent years, and continues to be studied, improved and applied in a wide variety of scientific and engineering fields.

4.2. Discrete TLBO

The original version of TLBO is oriented to continuous problems, which means that the design variables usually are in the continuous domain. It must be taken into account that TSP is a combinatorial problem. This kind of problems involves finding a grouping, ordering, or assignment of a discrete, finite set of objects that satisfies a set of conditions. Candidate solutions are combinations of solution components that may be found during a solution approach but do not have to satisfy all the conditions. Solutions are candidate solutions that satisfy all the restrictions.

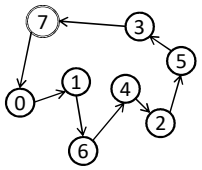
When applying TLBO to TSP, an in-depth modification must be made so as to adapt it to discrete, combinatorial problems. In case of TSP, some works can be found in the literature that proposed and approach to TLBO for discrete problems (DTLBO). In the current work, an implementation of the DTLBO based on the work developed in [32] has been performed, but several relevant modifications have been carried out.

4.2.1. Representation of individuals

Each individual is a sequence of city or urban locations or nodes, so each individual is a solution to be evaluated. As an example, if the problem deals with 8 urban locations (v_0, v_1... v_7) to be visited, an individual could be represented as depicted in Figure 1. In this figure, the individual solution consists of starting from urban node v_7 and finishing in node v_3 before returning to the starting node v_7, visiting the different nodes in the following order: $v_7 \rightarrow v_0 \rightarrow v_1 \rightarrow v_6 \rightarrow v_4 \rightarrow v_2 \rightarrow v_5 \rightarrow v_3$.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2020 doi:10.20944/preprints202010.0605.v1

In [32], the initial assignment of individuals is performed by random permutations of the urban nodes to be visited and the global population is divided into subpopulations so as to avoid becoming trapped in local minima, concretely four subpopulations are considered. In the current work, a modification is proposed which consists in using a greedy strategy when generating one of the individuals which form a subpopulation, while the others are randomly created; in this way, a sub-optimal solution can be counted on as a starting point, which can accelerate convergence towards an optimal solution.



Vector index	0	1	2	3	4	5	6	7
Path sequence	7	0	1	6	4	2	5	3

Figure 1. Representation of an individual solution.

4.2.2. DTLBO Teacher stage

A *Partial* Teacher is determined from each subpopulation which will be used so as to update learners in the subpopulation. The mean values within each subpopulation are also collected in a *Mean* individual that will also be used when updating each learner. A global *Teacher* is also determined as the best individual within the whole population.

It should be noted that, while the *Teacher* is a valid individual, the *Mean* individual could not be, as some locations could be repeated and others could not appear. As a consequence, a legality or feasibility operation must be performed on this *Mean* individual. This feasibility operation works as follows:

Step 1: Count the number of times each urban node appears. In the example shown in Figure 2, node labelled 3 appears 3 times, node 5 appears 2 times, and nodes labelled 0, 2, 6, and 4 appear once. This circumstance is expressed as a vector *TempA*, where, the number of times each urban node appears is written into the last cell where this node appeared; otherwise, the cell is set to empty (symbol −).

Step 2: Determine the urban nodes that do not appear. For the original individual, urban node labelled 1, 6, and 7 do not appear. This fact is expressed as a vector *TempB*. In this vector, the order number (index) of a node that does not appear is written into the associated cell, and the other cells are set to empty.

Step 3: Find out the last cell of a node that appears more than once. In the vector *TempC*, if a location appeared more than once, the order number (index) of this node is written into the last cell where this node appeared.

Step 4: Determine the cells of nodes that do not appear in the original individual. In the vector *TempC*, the order numbers (indices) of nodes that do not appear are written in the cells which contained an empty value, obtaining thus the resultant viable individual.

Crossover operation. The crossover operation is indicated by ⊗ symbol. There are four different crossover operations that can be selected so as to create a new individual:

$$Xnew(i) = X(i) \otimes Teacher \quad (5)$$

$$Xnew(i) = X(i) \otimes PartialTeacher(i)$$

$$Xnew(i) = X(i) \otimes Mean(i)$$

$$Xnew(i) = PartialTeacher(i) \otimes Mean(i)$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2020 doi:10.20944/preprints202010.0605.v1

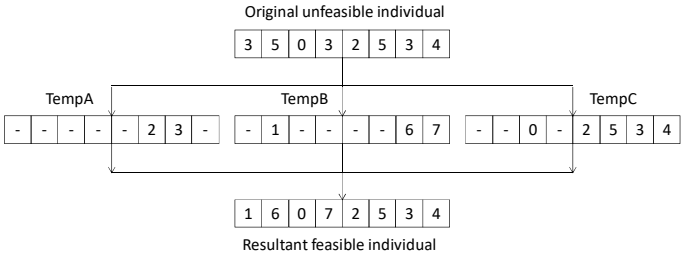


Figure 2. An example of the feasibility operation.

In the current work, a random selection of each one of the above crossover operations is performed each iteration and for each individual. This approach is different from the one used in [32], where each subpopulation was assigned a fixed expression for the crossover operation. In this way, a wider variety of individuals is achieved and, therefore, a new strategy is added to prevent a subpopulation from being trapped in a local minimum. By introducing this random selection, new solutions appear due to the fact that the crossover for each individual within a subpopulation can be different each one of the iterations.

The crossover operation works in the following way: given two individuals, A and B, a new individual A_c must be generated; a starting and an ending position in the order of visiting the urban locations are randomly selected, so that individual A_c replaces its components by the ones of individual B. Figure 3 shows an example of this operation. After a new individual is created, the feasibility operation must be performed on it.

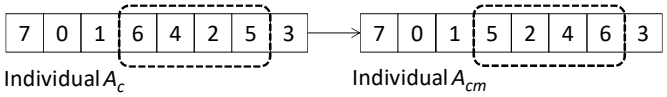


Figure 3. An example of the crossover operation.

Mutation operation. After the new individual is created by means of the crossover operation, a mutation operation is applied on it. Symbol Θ will be used so as to indicate the mutation operator. The mutation is performed as follows: a starting and an ending position in the order of visiting the urban locations are selected; then the elements are flipped between the two positions so as to generate a mutation in the individual.

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

As an example, given an individual A_c, a starting position 3 and an ending position 6 in the order of visiting the urban locations are randomly selected; the mutated individual A_{cm} is obtained by the mutation operation as shown in Figure 4:

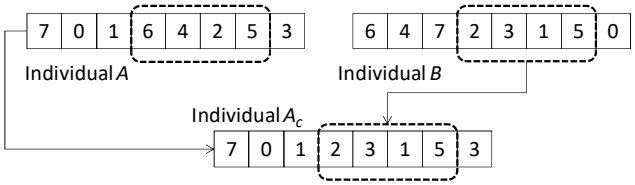


Figure 4. An example of the mutation operation.

4.2.3 DTLBO Learner stage

After the teacher stage, the learner stage of DTLBO is carried out. In a similar way as the original TLBO, for each individual i a contestant k is randomly selected within its subpopulation. The current learner individual X_i is updated as follows:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2020 doi:10.20944/preprints202010.0605.v1

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

where ⊗ represents the crossover operation, working in the same way as the crossover operation in the teacher stage.

Once the crossover operation is carried out, the feasibility filtering is performed on the new individual. Then the mutation operation is carried out just the same way as in the teacher stage.

In [32], DTLBO is compared to ACO, ABC, PSO (*Particle Swarm Optimization*), and GA (*Genetic Algorithm*) when coping with different TSP instances; conclusions remark that DTLBO is better than the above mentioned algorithms in most cases, and only ACO and ABC achieve similar or slightly better results in very few instances; the conclusions also highlight that the performance of DTLBO decreases when dealing with very large cities (with regard to the number of nodes); therefore, it is suggested that efforts should be made to focus on ways to improve DTLBO performance for TSP problems with large-scale cities.

5. Parallel TLBO implementation on GPU

As explained above, applying a series of modifications to the TLBO algorithm can achieve great results in discrete combinatorial problems, in this case in the TSP problem. Very satisfying results are achieved compared to other algorithms [32].

The modifications made on the original TLBO add more complexity and computational cost to the different stages and, therefore, the iterations of the DTLBO are much more expensive than those of the original TLBO, which penalizes the execution time. To minimize the impact of these changes and the extra cost of computing on the performance of the algorithm, a parallel version of the algorithm has been implemented using a CUDA architecture on GPU platform. While 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 substantial improvement in performance over sequential solutions using this parallel architecture. Some research works can be found that try improving metaheuristic methods applied to graph search problems by means of parallel implementations [33].

The initial and fundamental step when proceeding with the parallelization of the algorithm is to create a correct design of the memory structure and the execution flow in order to minimize the global thread locks. GPU memory is a resource that, if mismanaged, can negatively impact execution times because transfer operations between memory levels within the GPU are time-consuming and, therefore, must be minimized.

5.1. Design of the memory organization

The first step when proceeding with paralleling the algorithm is to conceive a correct design of the memory structure and the execution flow so as to minimize the blockage of the different threads executing in parallel within the GPU. The mismanagement of GPU memory can highly increase the computation time because transfer operations between the different GPU memory levels are very time-consuming. Thus, the GPU memory levels will be organized to manage different kind of information (thread, subpopulation, and global levels) as depicted in Figure 5.

Global memory. Two main blocks of information are stored in the global memory: on the one hand, an array where all the necessary points for the generation of the individuals of the populations in each block are stored; on the other hand, an array with all the pre-calculated distances between points; and finally the best individual of all the populations together with its evaluation (global *Teacher*) in each iteration in a global way to be used by the whole population. Much of the information stored in the global memory will be read-only and will be used to avoid distance calculations within the evaluation of each individual and to save memory in the individuals of each subpopulation; only the best evaluation (global *Teacher*) will be written and modified if necessary each iteration.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2020 doi:10.20944/preprints202010.0605.v1

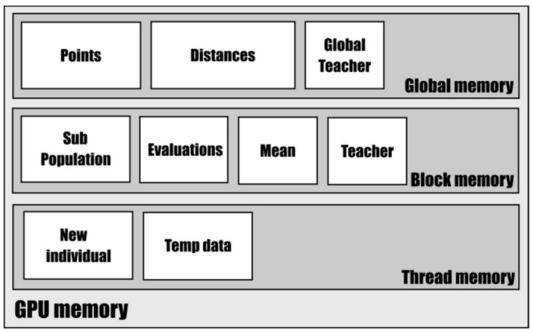


Figure 5. Organization of the CUDA memory.

Block memory. The information shared by a subpopulation is stored in the block memory. The memory is organized in a matrix where each row represents an individual and each column stores the index of an urban location; an extra column stores the individual's solution to avoid repeating the evaluation.

Thread memory. Some variables are stored in the thread local memory. All these variables are used for the required calculations within each phase of the algorithm and are updated each one of the iterations. This memory is private to each thread and is not shared with the rest of the population.

5.2. Execution flow

On the other hand, an execution flow has been designed with the intention of minimizing the blockages of the threads at the time of synchronization during the execution of the algorithm (*syncthreads*). TLBO includes a series of stages that force the threads to be synchronized in order to obtain common information for the whole population, such as the mean individual and the best individual (*Teacher*). In addition, to obtain this information, the reduction technique will be applied to minimize the iterations required to obtain these values from the population. In order to parallelize the execution of the algorithm, each thread will be used as an individual of the population; the first thread (indexed as 0) from each subpopulation is in charge of performing the operations on local memory, while the first thread from the first population is in charge of performing the required operations on global memory. This is depicted in Figure 6. In this way, conflicts and delays when accessing the different memory levels are avoided.

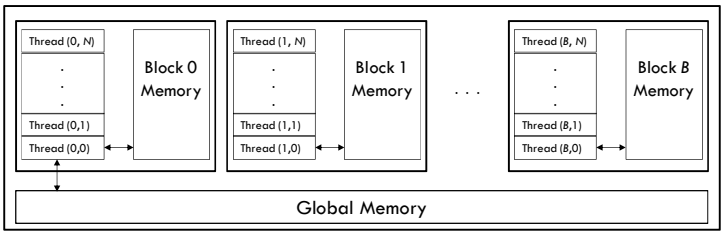


Figure 6. Subpopulations organization and memory accesses.

6. Experimentation

In the experiments, the comparison is focused on the computation time when solving four well-documented TSP problems from the TSPLIB library [34]. A comparison is made between a sequential implementation and a manycore GPU implementation using CUDA. The hardware used for the experimentation is a Pentium i7 processor at 3.2GHz with an NVIDIA GeForce 1060p graphics card, 6 GB GPU RAM and 32 GB DDR4 RAM. The CUDA platform version was 9.2. Different scenarios are defined for each problem by modifying the population sizes (64 and 128 individuals) and the number of iterations to run the algorithm (1,000, 5,000 and 10,000 iterations).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2020 doi:10.20944/preprints202010.0605.v1

The problems or cities from TSPLIB which are used for experimentation are 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 to 14 show the results with regard to the CPU and GPU average time in milliseconds of 10 blocks of 1,000 runs, each with a PC clean reset. Results demonstrate that the GPU parallel implementation of DTLBO improves performance up to 6x in case of a large number of individuals and iterations. DTLBO reached the optimal path in Att48 (distance = 33,523); in case of Berlin52, the difference obtained with regard to the optimal was only 2 (7,544 versus 7,542); in case of the problems with a higher number of urban nodes, DTBLO obtained 6,336 in Ch130 (optimal = 6,110), and 552.63 in Eil76 (optimal = 538). The paths obtained for these four real urban scenarios are shown in Figures 15 to 18.

Given the characteristics of the GPU and the possibility of processing a large number of threads per block, it can be seen that the increase in population does not have as much time penalty as the CPU, being the GPU times only affected by the number of iterations. In addition, the GPU block architecture is well suited for algorithms with subpopulations, since they can be placed on different GPU blocks and, therefore, run in parallel if the number of subpopulations allows for it. For real-world applications, different problems can be run on the same GPU, maximizing GPU utilization and minimizing system response time.

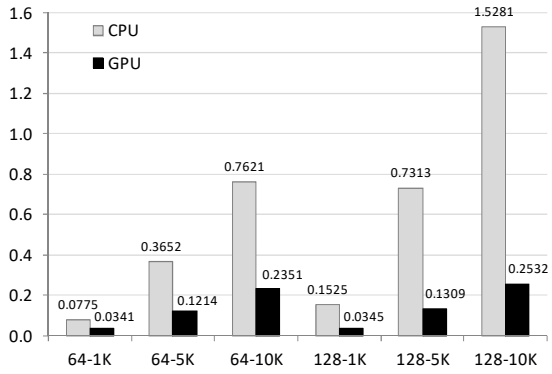


Figure 7. Results from Berlin130 with regard to different number of individuals (64 and 128) and iterations (1000, 5000 and 10000). CPU and GPU time.

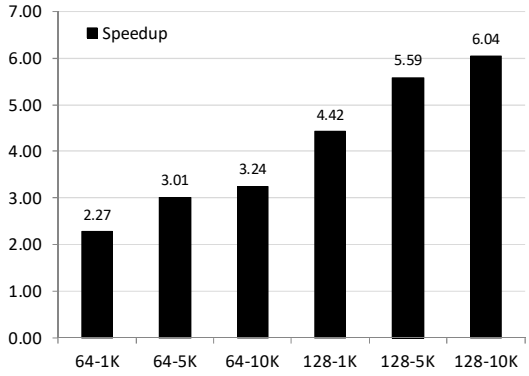


Figure 8. Results from Berlin130 with regard to different number of individuals (64 and 128) and iterations (1000, 5000 and 10000) Corresponding speedup.

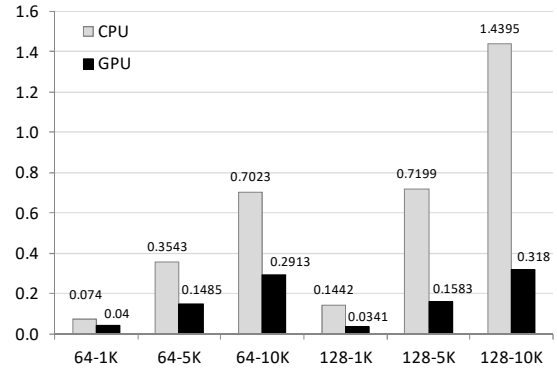


Figure 9. Results from Att48 with regard to different number of individuals (64 and 128) and iterations (1000, 5000 and 10000). CPU and GPU time.

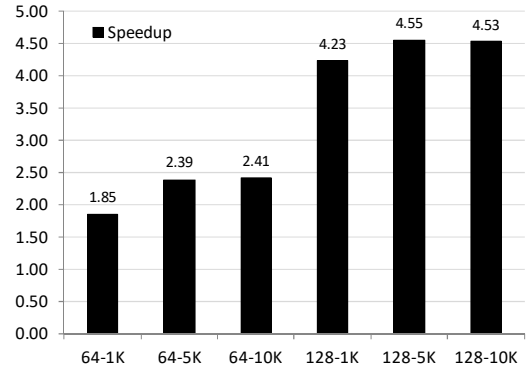


Figure 10. Results from Att48 with regard to different number of individuals (64 and 128) and iterations (1000, 5000 and 10000). Corresponding speedup.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2020 doi:10.20944/preprints202010.0605.v1

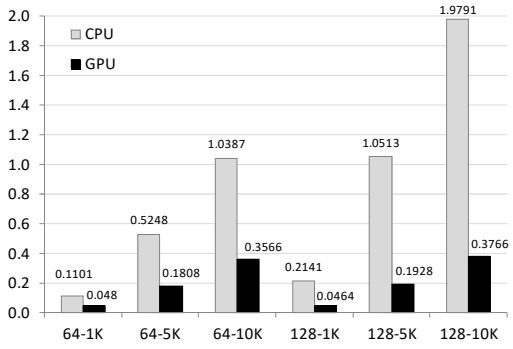


Figure 11. Results from Eil76 with regard to different number of individuals (64 and 128) and iterations (1000, 5000 and 10000). CPU and GPU time.

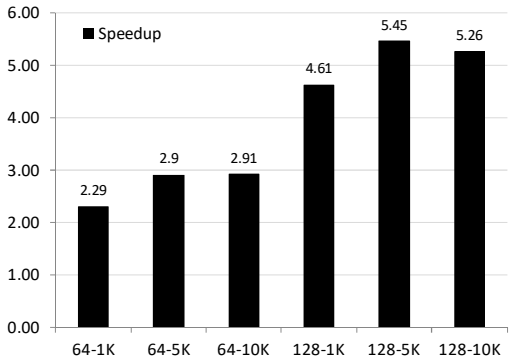


Figure 12. Results from Eil76 with regard to different number of individuals (64 and 128) and iterations (1000, 5000 and 10000). Corresponding speedup.

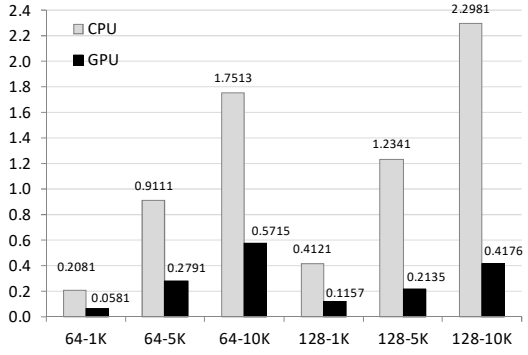


Figure 13. Results from Ch130 with regard to different number of individuals (64 and 128) and iterations (1000, 5000 and 10000). CPU and GPU time.

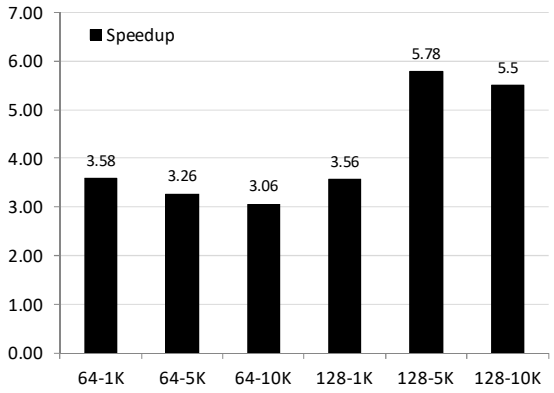


Figure 14. Results from Ch130 with regard to different number of individuals (64 and 128) and iterations (1000, 5000 and 10000). Corresponding speedup.

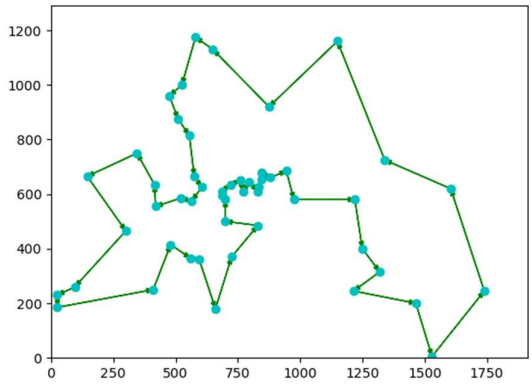


Figure 15. DLTBO solution for Berlin52.

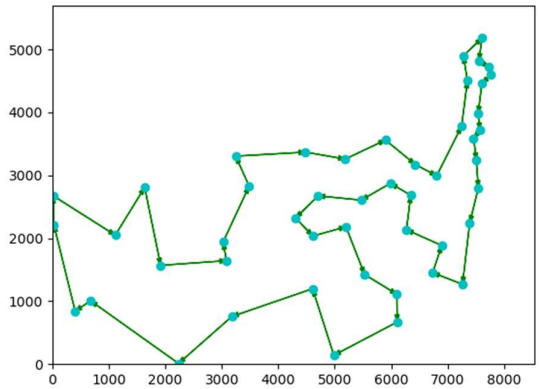


Figure 16. DLTBO solution for Att48.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2020 doi:10.20944/preprints202010.0605.v1

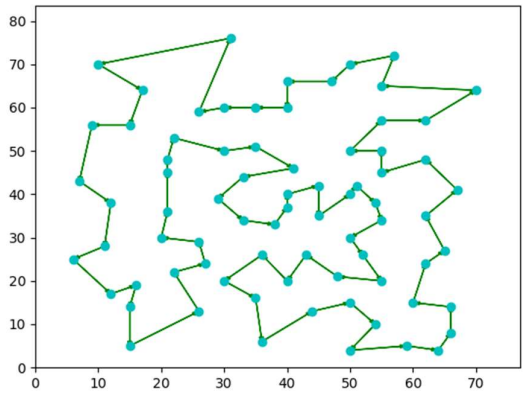


Figure 17. DTLBO solution for Eil76

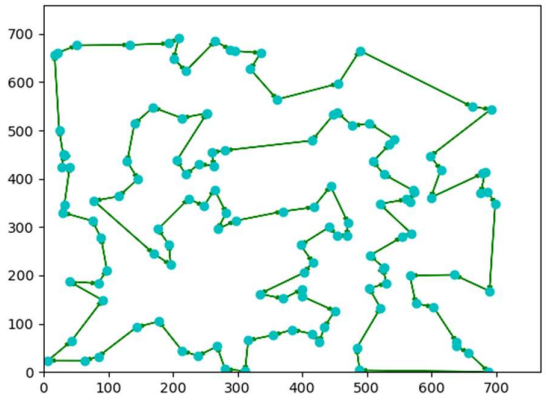


Figure 18. DTLBO solution for Ch130

7. Conclusions

In this paper, a parallel implementation of the discrete Teaching-Learning-Based Optimization algorithm applied to the Traveling Salesman Problem using a manycore GPU approach to improve performance is proposed. This algorithm has proven in previous works to be a good choice for solving TSP problems, but its biggest drawback is its complexity and computational cost when compared to other metaheuristic optimization algorithms. The parallel implementation using manycore GPU proposed and developed in this research work has substantially improved the speed of the algorithm, obtaining important speedups with regard to a sequential implementation; in case of a large number of individuals and iterations, the speedup reaches 6x. These results make the algorithm suitable to be applied in problems where the execution time is one of the determining factors, in special when the urban nodes that must be visited following an optimized path is very high. The implementation developed could still be improved in several parts trying to improve and emphasize the thread blocks to optimize the use of GPU resources.

In addition, although in this research work parallel implementations of the algorithm on CUDA have been carried out on a desktop computer, these algorithms can be executed in embedded systems inside cars, since there are already some cars with NVIDIA chipsets and support for CUDA, which are already 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 and it can be troublesome to run processes on the CPU that make intensive use of it, since the CPU must manage other subsystems in the vehicle.

Author Contributions: H. Rico-Garcia: Conceptualization, Methodology, Software, Validation, Investigation, Writing – original draft, Writing - review & editing. **J.L. Sanchez-Romero:** Conceptualization, Methodology, Software, Validation, Investigation, Writing – original draft, Writing – review & editing, Supervision, Project administration. **A. Jimeno-Morenilla:** Validation, Investigation, Writing – original draft, Writing -review & editing, Supervision, Project administration, Funding acquisition. **H. Migallon:** Validation, Investigation, Writing – original draft, Writing - review & editing, Supervision, Funding acquisition.

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.

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: Torre T., Braccini A., Spinelli R. (eds) Empowering Organizations. Lecture Notes in Information Systems and Organisation 11. Springer, Cham (2016).
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: 2016 International Conference on Emerging Technological Trends (ICETT). IEEE (2016).
3. Transport Statistics Great Britain: 2019 summary. Department for Transport. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/870647/tsgb-2019.pdf, last accessed 2020/08/04.
4. American Public Transportation Association. Public Transportation Facts. https://www.apta.com/news-publications/public-transportation-facts, last accessed 2020/08/04.
5. Fernandez-Ares, A., et al.: Studying real traffic and mobility scenarios for a Smart City using a new monitoring and tracking system. Future Generation Computer Systems 76, 163-179 (2017).
6. Sendra, S., et al.: Collaborative LoRa-Based Sensor Network for Pollution Monitoring in Smart Cities. In: 2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC). IEEE (2019).
7. Kazmi, A., Tragos, E., Serrano, M.: Underpinning IoT for road traffic noise management in smart cities. In: 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). IEEE (2018).
8. Kök, İ., Şimşek, M.U., Özdemir, S.: A deep learning model for air quality prediction in smart cities. In: 2017 IEEE International Conference on Big Data (Big Data). IEEE (2017).
9. Jabbarpour, M.R., Nabaei, A., Zarrabi, H.: Intelligent Guardrails: an IoT application for vehicle traffic congestion reduction in smart city. In: 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). IEEE (2016).
10. Singh, D., Vishnu, C., Mohan, C.K.: Visual Big Data analytics for traffic monitoring in smart city. In: 2016 15th IEEE international conference on machine learning and applications (ICMLA). IEEE (2016).
11. Pawłowicz, B., Salach, M., Trybus, B.: Smart city traffic monitoring system based on 5G cellular network, RFID and machine learning. In KKIO Software Engineering Conference (pp. 151-165). Springer, Cham (2018).
12. Rathore, M.M., et al.: Exploiting IoT and big data analytics: Defining smart digital city using real-time urban data. Sustainable cities and society 40, 600-610 (2018).
13. Behnke, M., Kirschstein, T.: The impact of path selection on GHG emissions in city logistics. Transportation Research Part E: Logistics and Transportation Review 106, 320-336 (2017).
14. Ehmke, J.F., Campbell, A.M., Thomas, B.W.: Data-driven approaches for emissions-minimized paths in urban areas. Computers & Operations Research 67, 34-47 (2016).
15. Suzuki, Y.: A dual-objective metaheuristic approach to solve practical pollution routing problem. International Journal of Production Economics 176, 143-153 (2016).
16. Ehmke, J.F., Campbell, A.M., Thomas, B.W.: Vehicle routing to minimize time-dependent emissions in urban areas. European Journal of Operational Research 251(2), 478-494 (2016).
17. Kramer, R., et al.: A matheuristic approach for the pollution-routing problem. European Journal of Operational Research 243(2), 523-539 (2015).
18. Rego, C., et al.: Traveling salesman problem heuristics: Leading methods, implementations and latest advances. European Journal of Operational Research 211(3), 426-441 (2011).
19. Ilie, S.V.: Survey on distributed approaches to swarm intelligence for graph search problems. Annals of the University of Craiova-Mathematics and Computer Science Series 41(2), 251-270 (2014).
20. Karaboga, D., Gorkemli, B.: Solving traveling salesman problem by using combinatorial artificial bee colony algorithms. International Journal on Artificial Intelligence Tools 28(01), 1950004 (2019).
21. 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. Transportation Research Part D: Transport and Environment 57, 422-457 (2017).
22. Gan, R., et al.: Improved ant colony optimization algorithm for the traveling salesman problems. Journal of Systems Engineering and Electronics 21(2), 329-333 (2010).

23. Shokouhifar, M., Sabet, S.: PMACO: A pheromone-mutation based ant colony optimization for traveling salesman problem. In: 2012 International Symposium on Innovations in Intelligent Systems and Applications. IEEE (2012).
24. Bai, J., et al.: A model induced max-min ant colony optimization for asymmetric traveling salesman problem. Applied Soft Computing 13(3), 1365-1375 (2013).
25. Głąbowski, M., et al.: Shortest path problem solving based on ant colony optimization metaheuristic. Image Processing & Communications 17(1-2), 7-17 (2012).
26. Mahi, M., Baykan, O.K., Kodaz, H.: A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem. Applied Soft Computing 30, 484-490 (2015).
27. Rao, R.V, Savsani, V., Vakharia, D.P.: Teaching–Learning-Based Optimization: a novel method for constrained mechanical design optimization problems, Computer-Aided Design 43(3), 303-315 (2011).
28. Rao, R.V., Patel, V.: An elitist Teaching-Learning-Based Optimization algorithm for solving complex constrained optimization problems. International Journal of Industrial Engineering Computations 3, 535–560 (2012).
29. Rao, R.V., Patel, V.: Comparative performance of an elitist Teaching-Learning-Based Optimization algorithm for solving unconstrained optimization problems. International Journal of Industrial Engineering Computations 4, 29–50 (2013).
30. Ebraheem, M., Jyothsna, T.R.: Comparative performance evaluation of Teaching Learning Based Optimization against genetic algorithm on benchmark functions. In: 2015 Power, Communication and Information Technology Conference (PCITC), 327-331 (2015).
31. Shah, S.R., Takmare, S.B.: A Review of Methodologies of TLBO Algorithm to Test the Performance of Benchmark Functions. Programmable Device Circuits and Systems 9(7), 141-145 (2017).
32. Wu, L., Zoua, F., Chen, D.: Discrete Teaching-Learning-Based Optimization Algorithm for Traveling Salesman Problems. In: MATEC Web of Conferences 128, 02022. EDP Sciences (2017).
33. Arnautović, M., et al.: Parallelization of the ant colony optimization for the shortest path problem using OpenMP and CUDA. In: 2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE (2013).
34. Universität Heidelberg. Institut für Informatik. TSPLIB. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/, last accessed 2020/08/04.