The Journal of Supercomputing

Solving travelling salesman problem using parallel repetitive nearest-neighbor algorithm on OTIS-Hypercube and OTIS-Mesh optoelectronic architectures

--Manuscript Draft--

Manuscript Number:	SUPE-D-17-00181
Full Title:	Solving travelling salesman problem using parallel repetitive nearest-neighbor algorithm on OTIS-Hypercube and OTIS-Mesh optoelectronic architectures
Article Type:	Manuscript
Keywords:	parallel heuristics algorithm; nearest-neighbor algorithm; travelling salesman problem; optoelectronic architecture; interconnection network; OTIS.
Abstract:	Over the past twenty years, researchers drew their attention to propose optoelectronic architectures for parallel systems, including Optical Transpose Interconnection System (OTIS) networks. On the other hand, there are limited attempts devoted to design parallel algorithms for important applications that could be mapped on such optoelectronic architectures. Thus, exploiting the attractive features of OTIS networks and investigating their performance in solving combinatorial optimization problems become a great necessity. In this paper, Parallel Repetitive Nearest-Neighbor (PRNN) algorithm for solving the travelling salesman problem (TSP) on OTIS-Hypercube and OTIS-Mesh optoelectronic architectures is presented. This algorithm has been evaluated analytically and by simulation in terms of number of communication steps, parallel run time, speedup, and efficiency on both optoelectronic architectures. Also, set of simulation runs were carried out on various number of cities. The simulation results attained almost near linear speedup and high efficiency.

Solving travelling salesman problem using parallel repetitive nearest-neighbor algorithm on OTIS-Hypercube and OTIS-Mesh optoelectronic architectures

Aryaf Al-Adwan, Basel A. Mahafzah*, Ahmad Sharieh

The University of Jordan,

King Abdullah II School for Information Technology,

Computer Science Department,

Amman 11942 – Jordan

Corresponding Author, e-mail: b.mahafzah@ju.edu.jo

Abstract Over the past twenty years, researchers drew their attention to propose optoelectronic architectures for parallel systems, including Optical Transpose Interconnection System (OTIS) networks. On the other hand, there are limited attempts devoted to design parallel algorithms for important applications that could be mapped on such optoelectronic architectures. Thus, exploiting the attractive features of OTIS networks and investigating their performance in solving combinatorial optimization problems become a great necessity. In this paper, Parallel Repetitive Nearest-Neighbor (PRNN) algorithm for solving the travelling salesman problem (TSP) on OTIS-Hypercube and OTIS-Mesh optoelectronic architectures is presented. This algorithm has been evaluated analytically and by simulation in terms of number of communication steps, parallel run time, speedup, and efficiency on both optoelectronic architectures. Also, set of simulation runs were carried out on various number of cities. The simulation results attained almost near linear speedup and high efficiency.

Keywords Parallel Heuristics Algorithm, Nearest-Neighbor Algorithm, Travelling Salesman Problem, Optoelectronic Architecture, Interconnection Network, OTIS

1. Introduction

With the rapid development of parallel systems, relying only on the speed of the processing elements is not satisfactory. The types and the ways of interconnecting these processing elements play great role in their performance too. This stimulates researchers for proposing hybrid interconnection networks that use both electronic and optical interconnection topologies, which utilize optical and electronic links. These interconnection networks are known as optoelectronic architectures or swapped interconnection networks [1].

One of the well-known optoelectronic architecture that gained attention recently is Optical Transpose Interconnection System (OTIS) [1], where the processors are connected to form groups of basic network; such as ring, mesh, hypercube, etc. The processors inside each group are connected using electronic links, while the groups themselves are connected to each other using

optical links. This emerges subsequent types of OTIS networks such as OTIS- Hypercube, OTIS-Mesh [2, 3], OTIS-Mesh-of-Trees [4], and OTIS-Hyper Hexa-Cell (OHHC) [5]. These optoelectronic architectures stimulate the researchers to develop parallel algorithms for basic operations that could be mapped efficiently on them. Such as sorting, routing, data accumulation, prefix sum, consecutive sum and matrix multiplications, all of them were implemented on OTIS-Mesh [6-8]. Other operations such as load balancing were implemented on OTIS-Hypercube [9]. However, very little attention was paid in exploiting the optical attractive feature of OTIS optoelectronic architectures in solving NP-hard optimization problems; such as Travelling Salesman Problem (TSP). Thus, TSP was chosen in this paper in order to evaluate the performance of OTIS optoelectronic architectures and investigate the usefulness of them. TSP is an optimization problem that stands out among the most challenging problems in operational research and computer science [10]. The problem can be described as a group of cities and a salesperson in a certain city has to visit all the remaining cities once and only once and return to the starting city with a minimum cost tour [10-13]. Exact algorithms evaluate all possible solutions to provide the optimal one exist for small number of cities, but they become inefficient when the number of cities is large. Accordingly, numerous heuristic algorithms were proposed to solve this problem to provide suboptimal solution rather than optimal one. One of those were the constructive methods that build a tour then stop when one solution is obtained such as nearest-neighbor algorithm [14, 15]. However, to the best of our knowledge there is no work that has been investigated the performance of the OTIS optoelectronic architecture in solving TSP. Thus, in this paper, we present Parallel Repetitive Nearest-Neighbor (PRNN) algorithm for solving TSP on OTIS-Hypercube and OTIS-Mesh optoelectronic architectures. This algorithm is evaluated analytically under the following performance metrics; number of communication steps, speedup, and efficiency on OTIS-Hypercube and OTIS-Mesh. Also, its performance was evaluated by simulation runs on both optoelectronic architectures.

The organization of the paper is as follows: Section 2 introduces a background on OTIS optoelectronic architectures, namely; OTIS-Hypercube and OTIS-Mesh. Section 3 illustrates the PRNN algorithm in solving TSP over OTIS-Hypercube and OTIS-Mesh. Section 4 presents analytical evaluation of this algorithm on both optoelectronic architectures. Sections 5 shows the simulation setup and results. Finally, Section 6 summarizes the overall work and presents some suggested future work.

2. OTIS optoelectronic architecture

OTIS optoelectronic architecture provides efficient connections with higher scalability and lower complexity among the connected nodes, in addition to utilizing the optical technology in sending and receiving data between the connected groups [1]. They consist of groups, where these groups are connected as a basic network; such as ring, mesh, hypercube, etc. or this groups can be hybrid networks such as Mesh-of-Trees, Hyper Hexa-Cell, etc. The communication links within each group are electronic and between groups are optical. It is called transpose (swapped) interconnection network because of its strategy in connecting the groups [1]. The connection between nodes in different groups is done by transposing node's and group's labels. For example, the third processor in the second group is connected by optical link with the second processor in the third group and so on [1, 16]. Two OTIS optoelectronic architectures with basic factor networks were chosen, OTIS-Hypercube and OTIS-Mesh, to solve TSP using PRNN algorithm and to

evaluate its performance in providing near optimal solutions with less computational time. The following subsections illustrate the structure of OTIS-Hypercube and OTIS-Mesh.

2.1 OTIS-Hypercube

OTIS-Hypercube consists of N^2 processors that are partitioned into N groups, and each group contains N processors interconnected as a hypercube of dimension d, which is equal to log N. For example, in 16 processors two-dimensional OTIS-Hypercube, there are four groups, and each group has four processors, as depicted in Fig. 1. The solid arrows represent electronic links among processors and the dashed arrows represent optical links among groups. A label of any processor inside OTIS-Hypercube must represent the address of the group and the address of the processor within this group. For example, processor 2 in group 1 must have the label (01, 10). Formally, processor (i, j) represents processor j in group i and this processor is connected to processor i in group j via optical link [2].

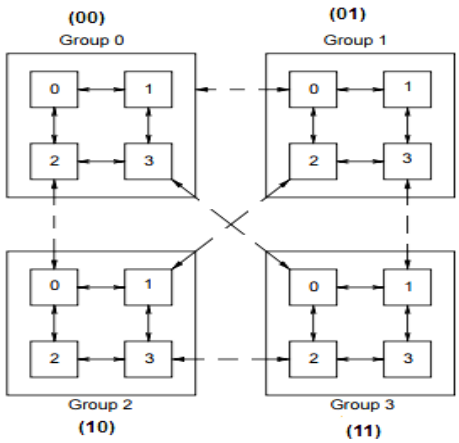

Fig. 1 Two-Dimensional OTIS-Hypercube [2]

2.2 OTIS-Mesh

OTIS-Mesh is another example of OTIS basic networks, which consists of N^2 processors that are partitioned into N groups, and each group contains N processors interconnected as a two-dimensional $\sqrt{N} \times \sqrt{N}$ mesh. The communication links within each group are electronic and between groups are optical. For example, in 81 processors OTIS-Mesh, there are 81 processors grouped into 9 groups, and each group has 9 processors, as shown in Fig. 2. Solid lines in Fig. 2, represent electronic links and dashed lines represent optical links, where processor (i, j) is connected to processor (j, i). For simplicity, only the optical links for group zero were showed in Fig. 2. It is important to mention that for 16 processors in OTIS-Mesh is similar to 16 processors in OTIS-Hypercube [2], as depicted in Fig. 1.

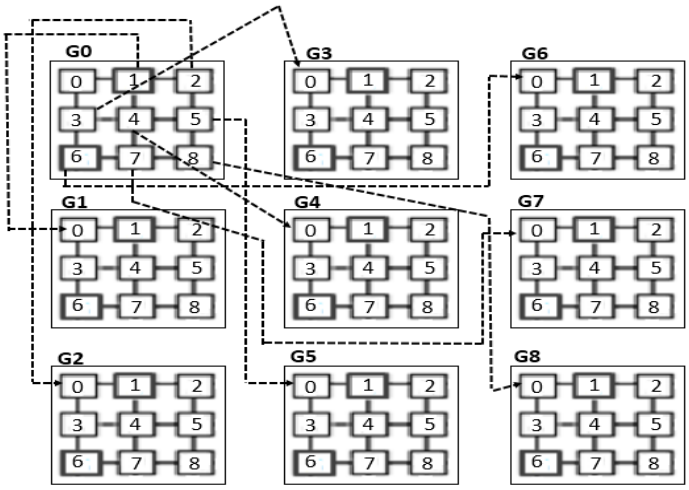

Fig. 2 OTIS-Mesh of nine groups and each group is 3×3 two-dimensional mesh

3. Solving TSP using PRNN algorithm

TSP involves iterative processes that exhaust the available resources, particularly when it is applied to a huge input size using a sequential machine. Solving small instances of TSP is not sufficient since large instances of this problem associated with various industrial and scientific applications. Thus, TSP should be solved by parallel algorithms that can perform the computations of large instances of cities in less time. In this section, a well-known TSP heuristic algorithm is selected to be parallelized over interconnections of interest. The algorithm along with its illustration followed by its analytical evaluation is presented first.

3.1 PRNN algorithm on OTIS-Hypercube

Tour construction heuristics were presented to obtain low cost good solutions in a reasonable time. They can be identified by building a solution through a sequence of steps based on immediate advantageous. These steps keep evolved until a valid solution is obtained. All tour construction heuristics stop when a solution is found without any improvement to get better solutions. Nearest-Neighbor (NN) is the simplest and well-known among TSP tour construction heuristics [14]. Obviously, its popularity resulted from its simplicity in implementation in addition to its ability to generate a good solution in a polynomial time. The sequential version of NN algorithm starts at a random city and traverses to the nearest city in a greedy way, as shown in Fig. 3.

Step1*: Pick a random city as a current vertex Cv.*
Step2*: Find the nearest unvisited city Nu.*
Step3*: Set the current vertex Cv as Nu.*
Step4*: Repeat Step 2 until there is no visited cities any more.*
Step5*: Link the starting city with the last one to form the tour.*

Fig. 3 The sequential nearest-neighbor algorithm [14]

The time complexity of this algorithm is O(n^2), since for each city among n cities the algorithm will search other n cities to find which city is the closest. However, NN algorithm can generate an approximate solution above 25% of the Held Karp lower bound [15]. Generally, the quality of solutions depends basically on the starting city. Therefore, another variation of nearest-neighbor algorithm includes repetitive execution of the algorithm on each city as a starting city to get better solution, but it requires intensive computations and time complexity of O(n^3). On the other hand, the architecture of OTIS interconnections enables us to adopt the Repetitive Nearest-Neighbor (RNN), where there are standalone processors connected together using electronic and optical links as a data media for high speed communication, and each processor has enough memory to perform different tasks in parallel. This adaptation will meet the purpose to obtain high quality solutions in less time.

The PRNN algorithm in this paper is designed based on the attractive features of the selected OTIS optoelectronic architectures; such as the iterative structure among groups, and the optical links existence between these groups. The algorithm is composed of four phases, namely: *load balancing* phase, *data distribution* phase, *local repetitive nearest-neighbor* phase and *data combining* phase. The Main Coordinator (*MC*), which is processor 00 in group G_0, is responsible for the load balancing phase. This phase is required, since partitioning the cities among processors yields to uneven number of cities assigned to processors. This is due to the fact that TSPLIB data set contains number of cities that is not power of two [17]. Consequently, we have introduced a load balancing algorithm, as illustrated in Fig. 4.

```
Algorithm: Load balancing algorithm
Input: Number of Cities C, Number of Processors P
Output: Number of Cities C is load balanced among Processors P
Main Coordinator (MC) performs the following:
1.    N_over_P = C / P
2.    extraCities = P× (N_over_P mod 1)
3.    for all processors P
4.        do
5.            Assign (N_over_P) cities for each processor
6.    If ((N_over_P) mod 2 != 0)
7.        then
8.              BalancedGroups= extraCities / numberOfProcessorsInsideGroup
9.              LastGroup= extraCities mod numberOfProcessorsInsideGroup
10.       for all extraCities
11.             do
12.                 Assign one extra city to each processor in the BalancedGroups
13.                 Assign one extra city to each processor in the LastGroup
```

Fig. 4 Load balancing algorithm

The load balancing phase is achieved by the *MC* processor via applying the load balancing algorithm (Fig. 4), which guarantees only one extra city for the balanced processors. This clearly can be demonstrated through lines 1-13 in Fig. 4. Lines (1-2) calculate the number of cities that must be handled by each processor and the number of extra cities that must be balanced between them. Lines (3-5) assign the calculated number of cities to each processor in the optoelectronic architecture. Obviously, as mentioned before, partitioning the total number of cities on the total number of processors will yield a remainder that serves as the number of extra cities. To alleviate

this problem, lines (6-9) calculate the entire number of groups that must be balanced and the number of processors in the last group that must be balanced too. Lines (10-13) will assign the extra cities to the processors in the balanced groups, then to the processors inside the last group. The time complexity of this algorithm is $\Theta(p)$, since the extra cities will not exceed the total number of processors p. At the end of this stage, *MC* processor generates the Allocated Cities Matrix (*ACM*) which indicates the set of cities for each processor to apply the sequential repetitive nearest-neighbor on; for example, processor P_{00} in Group 0 will apply the sequential repetitive nearest-neighbor on cities 0 through 8, as depicted in Fig. 5. Recalling the rest of the phases of the PRNN algorithm: *data distribution* phase, *local repetitive nearest-neighbor* phase and *data combining* phase, where they are illustrated in Fig. 6.

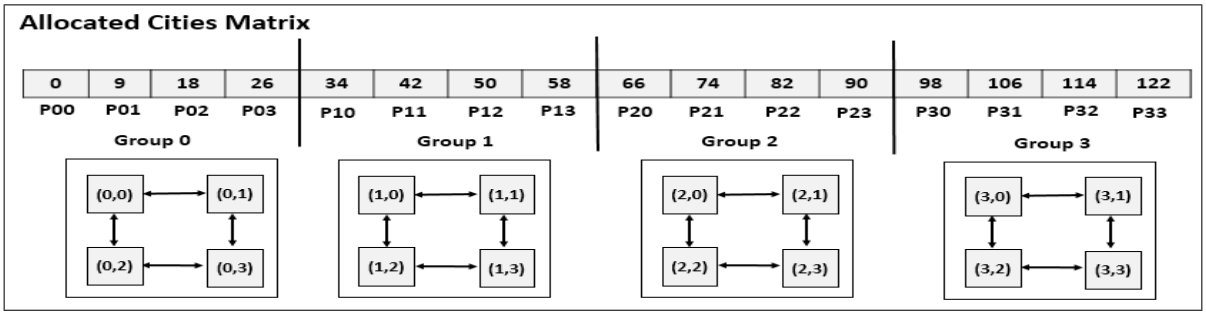

Fig. 5 *ACM* created by *MC* processor

Algorithm: Parallel Repetitive Nearest-Neighbor (PRNN) on OTIS-Hypercube
Input: Adjacency Distance Matrix DM for Graph G
Output: Shortest route among all cities

Phase 1: Load Balancing Phase
1. MC processor applies the load balancing algorithm (Fig. 4)
to balance the number of cities among all processors in OTIS-Hypercube
Phase 2: Data Distribution Phase
Electronic Main Group Distribution
*2. **for** each pair of processors differs only in the d^{th} bit position and belongs to the same group **do***
4. Each processor routes DM and ACM internally utilizing the hamming distance through the electrical link
5. MC processor stops the process of main group distribution and announces to start the next step
Optical Distribution of Data
*6. **for** all processors in Main Group (MG) that received DM, **do** in parallel*
7. Send DM to each Group Coordinator (GC) processor of the connected group via optical link
Inter Group Distribution of Data
*8. **for** all Group Coordinators (GCs), **do** in parallel*
9. *Repeat Steps 2-4*
Phase 3: Local Repetitive Nearest-Neighbor Phase
*10. **for** all processors in OTIS-Hypercube, **do** in parallel*
11. Apply sequential nearest-neighbor algorithm on each city in its set of cities
Phase 4: Data Combining Phase
Inter Group Data Combining
*12. **for** each pair of processors differs only in the d^{th} bit position and belongs to the same group, **do** in parallel*
14. Each processor routes its Route Matrix (RM) internally utilizing the hamming distance through the electrical link
Optical Data Combining
*15. **for** all Group Coordinators (GCs), **do** in parallel*
16. Send the Group Route Matrices (GRMs) via optical links to the main group
Main Group Data Combining
17. Apply the steps (12-14) assuming that you have only one group G_0
18. MC processor combines the collected group route matrices and finds the minimum route and its cost

Fig. 6 PRNN algorithm on OTIS-Hypercube

The PRNN algorithm on OTIS-Hypercube in Fig. 6, is illustrated in more details as follows:

Phase 2: Data Distribution Phase

The Main Group (*MG*) is group 0 in OTIS-Hypercube. It contains the *MC* processor, which is processor <0, 0>. Similarly, each group contains *Group Coordinator* (*GC*) processor which is the processor that connects the group G_i with the main group via optical link. This processor has label <*j*, 0>. For example, *GC* processor with label<2, 0> is connected with processor 2 at group 0 (Main Group) via optical link. Now, assume that the Distance Matrix (*DM*) as an array of size n^2, where *n* is the number of cities, is stored on *MC*. Then, the distribution phase is composed of three steps as follows:

I. *Electronic Main Group Distribution* (lines 2-5 in Fig. 6): *MC* processor starts the process by balancing the total number of cities among the total number of processors based on the load balancing algorithm, as shown in Fig. 4. According to this algorithm, each processor will obtain a set of cites, where the sequential nearest-neighbor algorithm is applied *N/p* times based on considering different city as a starting city each time. Since *DM and ACM* are located at *MC* processor, then it is responsible to send a copy of the *DM* and *ACM* to all other processors using one to all broadcast. Initially, *MC* processor has *DM* and *ACM* and at the termination of this broadcasting there will be P_G copies of *DM and ACM*, each copy belongs to each processor in G_0, where P_G is the number of processors in each group. The communication starts along the highest dimension which can be specified by the Most Significant Bit (MSB) of the binary representation of the processor's label, and continues for each lower dimension, as shown in Fig. 7, through steps 1, 2, 3 and 4. This will take $log\ P_G$ electronic moves. Then, all the nodes that receives *DM and ACM* in *MG* start the optical distribution of data as shown in the next step.

II. *Optical Distribution of Data* (lines 6-7 in Fig. 6): All nodes in G_0 (in parallel) start sending *DM and ACM* through the optical links to their corresponding processors in other groups. This will require 1 OTIS move. For example, in Fig. 7 (step 5), processor 9 in group G_0 will send *DM* to processor 0 in group G_9 via optical link.

III. *Inter Group Distribution of Data* (lines 8-9 in Fig. 6): Each *GC* processor in each group in OTIS-Hypercube except G_0 will repeat in parallel the task of *MC* processor that was done in electronic main group distribution, in which the *GC* processor will resend *DM and ACM* to each processor in its group using the hamming distance, as shown in Fig. 7, through steps 1, 2, 3 and 4. Again, this will take *log p* electronic moves.

Finally, at the end of this phase, *p* copies of *DM and ACM* are distributed to all processors in all groups.

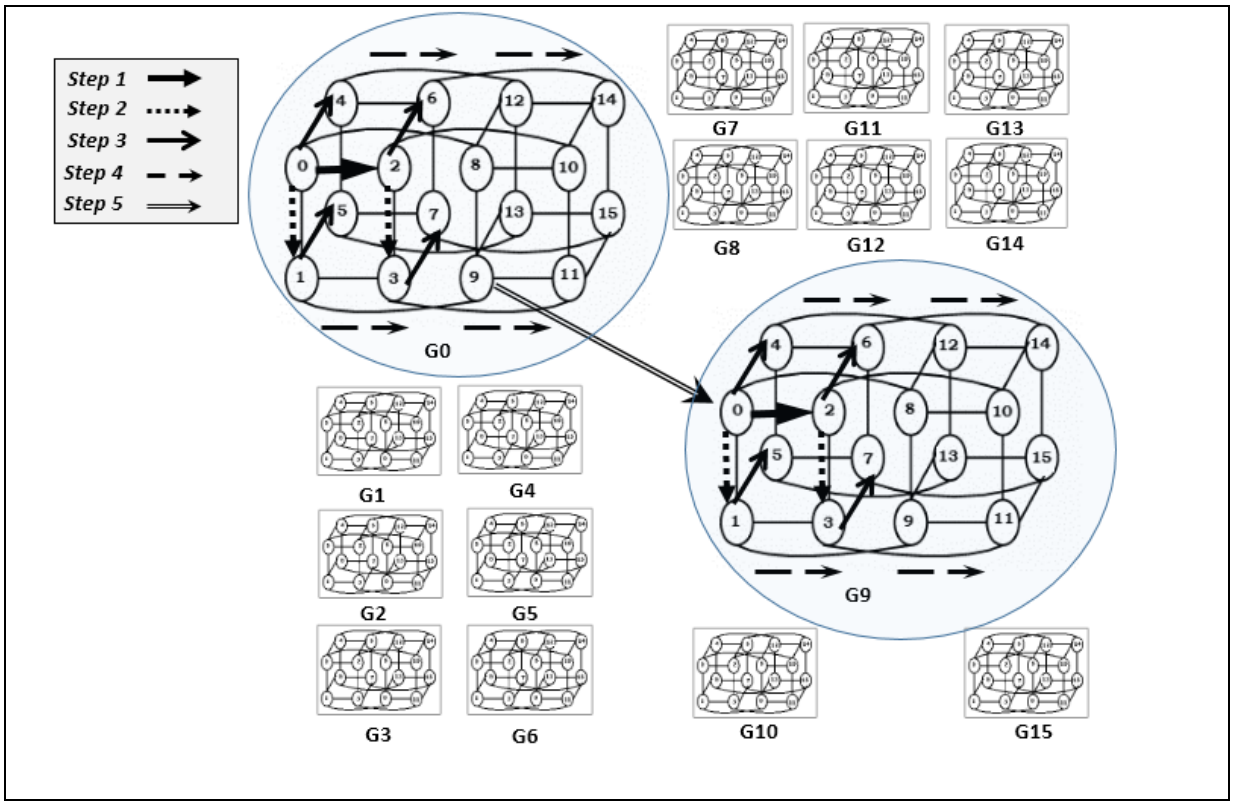

Fig. 7 Distribution of Distance Matrix (*DM*) on OTIS-Hypercube

Phase 3: Local Repetitive Nearest-Neighbor Phase *(lines 10-11 in Fig. 6)*

I. All processors in OTIS-Hypercube, in parallel, will apply the sequential *nearest-neighbor* algorithm on each city in the set of cities that belongs to each processor. The algorithm will be applied many times based on considering different city as a starting city every time, resulting in different routes for each starting city stored in a Route Matrix (*RM*), such that each processor will have its own *RM*.

Phase 4: Data Combining Phase

Data combining phase is done by overturning the order of steps in the distribution phase as follows:

I. *Inter Group Data Combining* (lines 12-14 in Fig. 6). The aim of this step is to combine all the *RMs* from all the processors in all groups to their associated *GC* processors via electronic links. Utilizing the hamming distance, a gather schema will be applied to combine *RMs*. This step will be performed in parallel to route the *RM* of each processor to all *GCs* processors in OTIS-Hypercube. During each communication step, each processor will receive the *RM* of its directly connected processor and will combine it with its own *RM* to be forward in the next communication step. The process continues in similar way until each *GC* processor in each group has gathered the collected *RMs* in one array called Group Route Matrix (*GRM*) which will be sent through the optical link. Note that the size of the communicated message will be enlarged based on the size of combined matrices.

II. *Optical Data Combining* (lines 15-16 in Fig. 6). All *GCs* processors in whole OTIS-Hypercube, except *GC* processor in G_0, will send their *GRMs* via optical links to their corresponding processors in the main group G_0.

III. *Main Group Data Combining* (lines 17-18 in Fig. 6). Each processor will combine the collected *GRM* with its own *RM* and sends it to its neighbor that differs in d^{th} bit position along the lower dimensions in $log\ P_G$ steps. At the end of this step, *MC* processor collects the *GRMs* and combines it with its own, in order to find the minimum route among all the collected routes.

3.2 PRNN algorithm on OTIS-Mesh

As illustrated previously PRNN algorithm is composed of four phases. The implementation of these phases over OTIS-Mesh is demonstrated in Fig. 8. Both load balancing and local repetitive nearest-neighbor phases are identical in OTIS-Mesh and OTIS-Hypercube. Whereas, distribution and combining phases differ from one interconnection to another based on the topological structure of these optoelectronic architectures. Thus, only the distribution and combining phases will be illustrated in more details in this section.

Phase 2: Data Distribution Phase

The distribution phase is composed of three steps as follows:

I. *Electronic Main Group Distribution* (lines 2- 6 in Fig. 8): *MC* processor must send a copy of the *DM* and *ACM* to all other processors using one to all broadcast. The communication will be accomplished in two phases. The first phase is *row-wise* phase, where *MC* processor will perform one to all broadcast to other $\sqrt{P_G} - 1$ processors in the same row. The second phase is *column-wise* phase, where each processor received *DM* will start one to all broadcast to all processors in its corresponding column through electrical links. At the end of this phase P_G copies of *DM* and *ACM* will be sent to P_G processors in G_0, and this requires $2\sqrt{P_G}$-1 communication steps. Then, all the nodes that receives *DM* and *ACM* in *MG* start the optical distribution of data as shown in the next step.

II. *Optical Distribution of Data* (lines 7-8 in Fig. 8): All nodes in G_0 (in parallel) start sending *DM* and *ACM* through the optical links to their corresponding processors in other groups. This will require one OTIS move.

III. *Inter Group Distribution of Data* (lines 9- 10 in Fig. 8): Each *GC* processor in each group in OTIS-Mesh except G_0 will repeat in parallel the task of *MC* processor that was done in electronic main group distribution, in which the *GC* processor will resend *DM* and *ACM* to each processor in its group through *row-wise* and *column-wise* phases, as have been done in electronic main group distribution.

Algorithm: Parallel Repetitive Nearest Neighbor (PRNN) on OTIS-Mesh
Input: Adjacency Distance Matrix DM for Graph G
Output: Shortest route among all cities

Phase 1: Load Balancing Phase
1. MC processor applies the load balancing algorithm (Fig. 4)
to balance the number of cities among all processors in OTIS-Mesh

Phase 2: Data Distribution Phase
Electronic Main Group Distribution
2. MC processor performs one-to-all broadcast to all processors in G_0 located in the same row
*3. **for** each processor in the same row of MC processor in G_0*
*4. **do***
5. route DM and *ACM using one-to-all broadcast to processors in its corresponding column through the electrical links*
6. MC processor stops the process of main group distribution and announces to start the next step
Optical Distribution of Data
*7. **for** all processors in MG that received DM, **do** in parallel*
8. Send DM to the Group Coordinator (*GC*) *processors of the connected groups via optical links*
Inter Group Distribution of Data
*9. **for** all Group Coordinators* (*GCs*) *processors, **do** in parallel*
10. Repeat Steps 2-5

Phase 3: Local Repetitive Nearest-Neighbor Phase
*11. **for** all processors in OTIS-Mesh, **do** in parallel*
12. Apply sequential nearest-neighbor algorithm on each city in its set of cities

Phase 4: Data Combining Phase
Inter Group Data Combining
*13. **for** each processor belongs to the same group and located in row number ($\sqrt{p}$-1)*
*14. **do** in parallel*
15. Each processor routes its Route Matrix (*RM*) *to processors in its corresponding column through the electrical links via gather operation.*
16. Each processor belongs to the same row with the GC processor performs gather operation
Optical Data Combining
*17. **for** all Group Coordinators* (*GCs*) *processors, **do** in parallel*
18. Send RM via optical links to the main group
Main Group Data Combining
19. Apply the steps (13-16), assuming that you have only one group G_0
20. MC processor combines the collected Group Route Matrices (*GRMs*) *and finds the minimum route and its cost*

Fig. 8 PRNN algorithm on OTIS-Mesh

Phase 4: Data Combining Phase

Data combining phase is done by overturning the order of steps in the distribution phase as follows:

I. *Inter Group Data Combining* (lines 13-16 in Fig. 8). A gather schema will be applied to combine *RMs*. This step will be performed in parallel to route the *RM* of each processor to the *GCs* processors in the OTIS-Mesh. In *column-wise* phase, each processor in row number $\sqrt{P_G}$-1 will perform a gather operation to sends its own *RM* to its directly connected processor in the same column. In the next communication step, each processor in each column, combines its own *RM* with the received one and forwards the concatenated *RMs.* This process continues,

until all processors in the same row with *GC* processor received the *RMs*. In *row-wise* phase, each processor in the same row with the *GC* processor will combine the received *RMs* and forward them to the next processor in the same row. Now, *GC* processor will combine the collected *RMs* in one array called Group Route Matrix (*GRM*) that will be sent through the optical links. Note that the size of the communicated message will be enlarged based on the size of the combined matrices.

II. *Optical Data Combining* (lines 17-18 in Fig. 8). All *GCs* processors in OTIS-Mesh, except *GC* processor in G_0, will send their *GRM* via optical links to their corresponding processors in the main group G_0.

III. *Main Group Data Combining* (lines 19-20 in Fig. 8). Each processor will combine the collected *GRM* with its own *RM* and sends it to its neighbor *column-wise* and then *row-wise* as illustrated in inter group data combining phase. At the end of this step, *MC* processor collects the whole *GRM* and combines it with its own, in order to find the minimum route among all the collected routes.

4. Analytical evaluation

This section provides the analytical evaluation of the PRNN algorithm on OTIS-Hypercube and OTIS-Mesh optoelectronic architectures. The performance metrics are used to evaluate the algorithm are time complexity, speedup and efficiency.

4.1 Analytical evaluation of PRNN algorithm on OTIS-Hypercube

In this section, the PRNN algorithm on OTIS-Hypercube is evaluated analytically in terms of parallel time complexity, speedup and efficiency.

4.1.1 Parallel time complexity

The parallel run time equals to the total communication time plus the total computation time. The communication time can be measured as a number of communication steps that is required in both the distribution and combining phases. The time required in applying the sequential nearest-neighbor algorithm on a set of cities represents the computation time. Thus, the time complexity of PRNN algorithm is captured in *Theorem 1*.

Theorem 1:
The average-case time complexity of PRNN algorithm on OTIS-Hypercube is shown in Equation (1), where T is the time complexity, N is the number of cities, p is the number of processors, and d is the dimension of the hypercube.

$$T(N, p) = \Theta\left(p + \frac{N^3}{P} + N^2 \times d\right) \quad (1)$$

Proof:
The analytical evaluation of the parallel run time complexity for all phases of PRNN algorithm on OTIS-Hypercube is demonstrated by tracing the algorithm in Fig. 6, as shown in Table 1 (a)–(d).

Table 1 (a) Run time complexity of phase 1 (*Load balancing phase*)

Line 1	*MC* processor in G_0 executes the load balancing algorithm in Fig. 4. This execution will partition the cities among p processors; such that each processor will take N/p cities. The run time complexity of this algorithm is $\Theta(p)$, since the extra cities will not exceed the total number of processors p.

Table 1 (b) Run time complexity of phase 2 (*Data distribution phase*)

First step: Electronic main group distribution	
Lines 2-6	*MC* processor sends *DM* to all processors in G_0. This process requires d steps, where d is the dimension of the hypercube, which is $log\ P_G$. Thus, the overall run time complexity equals to $d \times t_s + tw_{elect} \times (N^2 + p) \times d$, where t_s is the startup time, tw_{elect} is the time for one-word transmission via electronic links, N^2 is the size of the *DM* and p is the size of the *ACM* which is equal to the number of processors in the optoelectronic network.
Second step: Optical distribution of data	
Lines 7-8	The run time complexity of optical distribution of data requires $t_s + tw_{optical} \times (N^2 + p)$, which is the time needed to transmit *DM* of size N^2 and *ACM* of size p through the optical links, where $tw_{optical}$ is the time for one-word transmission via optical links.
Third step: Inter group distribution of data	
Lines 9-10	In parallel, every *GC* processor in each group sends *DM* and *ACM* to all processors in its group. This process requires d steps, where d is the dimension of the hypercube. Thus, the overall time complexity equals to $d \times t_s + tw_{elect} \times (N^2 + p) \times d$.
The overall parallel run time complexity of phase 2 is: $= (d \times t_s + tw_{elect} \times (N^2 + p) \times d) + (t_s + tw_{optical} \times (N^2 + p)) +$ $(d \times t_s + tw_{elect} \times (N^2 + p) \times d) \approx \Theta(N^2 \times d)$.	

Table 1 (c) Run time complexity of phase 3 (*Local repetitive nearest-neighbor phase*)

Lines 11-12	All processors in OTIS-Hypercube apply sequentially the nearest-neighbor algorithm on each city belongs to the set of cities associated with each processor. This will require $\frac{N}{p} \times N^2$ time complexity, where N is the number of cities, p is the number of processors, and N^2 is the run time complexity of the sequential nearest-neighbor algorithm.

Table 1 (d) Run time complexity of phase 4 (*Data combining phase*)

First step: Inter group data combining	
Lines 13-16	Combining *RMs* will be performed in *log* P_G steps, where $P_G = \sqrt{p}$. Thus, the time complexity of this step is $\sum_{i=1}^{logP_G}(ts + tw_{elect} \times 2^{i-1} \times \frac{N}{p} \times N)$ that is equal to $ts \times logP_G + tw_{elect} \times (P_G - 1) \times \frac{N}{p} \times N$.
Second step: Optical data combining	
Lines 17-18	The time complexity of this step, where each *GC* processor will send its *GRM* to G_0 via optical links, is equal to: $ts + tw_{optical} \times \frac{N}{p} \times N \times P_G$, where $P_G = \sqrt{p}$.
Third step: Main group data combining	
Lines 19-20	All processors in G_0 will send the accumulated *RM* to the *MC* processor. This will require $\sum_{i=1}^{logP_G}(ts + tw_{elect} \times 2^{i-1} \times \frac{N}{p} \times N \times P_G + \frac{N}{p} \times N)$ that is equal to $ts \times logP_G + tw_{elect} \times (P_G - 1) \times \frac{N^2}{p} \times (P_G + 1)$
The overall time complexity of phase 4 is: $= (ts \times logP_G + tw_{elect} \times (P_G - 1) \times \frac{N}{p} \times N) + (ts + tw_{optical} \times \frac{N}{p} \times N \times P_G) + (ts \times logP_G + tw_{elect} \times (P_G - 1) \times \frac{N^2}{p} \times (P_G + 1)) \approx \Theta\left(N^2 - \frac{N^2}{p}\right)$, where p is equal to P_G^2.	

The overall *parallel run time complexity* of phases 1–4 is shown in Equation (2). Thus, Equation (2) can be reduced to Equation (3).

$$T(N, p) = \Theta(p) + \Theta(N^2 \times d) + \Theta\left(\frac{N}{p} \times N^2\right) + \Theta(N^2 - \frac{N^2}{p}) \quad (2)$$

$$T(N, p) \approx \Theta(p + \frac{N^3}{P} + N^2 \times d) \quad (3)$$

4.1.2 Speedup

The speedup is represented as the ratio between the execution time required to solve a given problem sequentially on a single processor over the execution time required to solve the same problem on parallel machine [18]. The speedup of the PRNN algorithm on OTIS-Hypercube is shown as in Equation (4).

$$S = \frac{N^3 \times p}{p^2 + N^3 + N^2 \times d \times p} \quad (4)$$

4.1.3 Efficiency

Efficiency can serve as a performance metric to measure how much the processors being utilized [18] in the optoelectronic architecture. It is the ratio between speedup and number of processor. Therefore, the efficiency of the PRNN algorithm on OTIS-Hypercube is shown in Equation (5).

$$E = \frac{N^3}{p^2 + N^3 + N^2 \times d \times p} \quad (5)$$

4.2 Analytical evaluation of PRNN algorithm on OTIS-Mesh

In this section, the PRNN algorithm on OTIS-Mesh is evaluated analytically in terms of parallel time complexity, speedup and efficiency.

4.2.1 Parallel time complexity

The parallel time complexity of PRNN algorithm on OTIS-Mesh is captured in *Theorem 2*.

Theorem 2:
The average-case time complexity of PRNN algorithm on OTIS-Mesh is shown in Equation (6), where T is the time complexity, N is the number of cities, p is the number of processors, and P_G is the number of processors in each group.

$$T(N, p) = \Theta\left(p + \frac{N^3}{P} + N^2 \times \sqrt{P_G}\right) \quad (6)$$

Proof:
The analytical evaluation of the parallel run time complexity for all phases of PRNN algorithm on OTIS-Mesh is demonstrated by tracing the algorithm in Fig. 8, as shown in Table 2 (a)–(d).

Table 2 (a) Run time complexity of phase I (*Load balancing phase*)

Line 1	The run time complexity of this algorithm is $\Theta(p)$, since the extra cities will not exceed the total number of processors p.

Table 2 (b) Run time complexity of phase 2 (*Data distribution phase*)

First step: Electronic main group distribution	
Lines 2-5	*MC* processor sends *DM* to all processors in G_0 *row-wise* and *column-wise*. This process requires $\sqrt{P_G} - 1$ steps in the *column-wise* phase and $\sqrt{P_G} - 1$ steps in the *row-wise* phase. Thus, the overall run time complexity equals to $2 \times (\sqrt{P_G} - 1) \times t_s + tw_{elect} \times (N^2 + p) \times 2 \times (\sqrt{P_G} - 1)$, where P_G is the number of processors inside each group, t_s is the startup time, $tw_{elect}$is the time for one-word transmission via electronic links, N^2 is the size of the *DM*, and p is the size of the *ACM* which is equal to the number of processors in the optoelectronic architecture.
Second step: Optical distribution of data	
Lines 6-7	The run time complexity of optical distribution of data requires $t_s + tw_{optical} \times (N^2 + p)$, which is the time needed to transmit *DM* of size N^2 and *ACM* of size p through the optical links, where $tw_{optical}$is the time for one-word transmission via optical links.
Third step: Inter group distribution of data	
Lines 8-9	In parallel, every *GC* processor in each group sends *DM* to all processors in its group. This process requires $\sqrt{P_G} - 1$ steps. Thus, the overall time complexity equals to $2 \times (\sqrt{P_G} - 1) \times t_s + tw_{elect} \times (N^2 + p) \times 2 \times (\sqrt{P_G} - 1)$.
The overall run time complexity of phase 2 is: $(2 \times (\sqrt{P_G} - 1) \times t_s + tw_{elect} \times (N^2 + p) \times 2 \times (\sqrt{P_G} - 1)) + (t_s + tw_{optical} \times (N^2 + p)) + (2 \times (\sqrt{P_G} - 1) \times t_s + tw_{elect} \times (N^2 + p) \times 2 \times (\sqrt{P_G} - 1)) \approx \Theta\ (N^2 \times \sqrt{P_G})$.	

Table 2 (c) Run time complexity of phase 3 (*Local repetitive nearest-neighbor phase*)

Lines 10-11	This will require $\frac{N}{p} \times N^2$ time complexity, where N is the number of cities, p is the number of processors, and N^2 is the run time complexity of the sequential nearest-neighbor algorithm.

Table 2 (d) Run time complexity of phase 4 (*Data combining phase*)

First step: Inter group data combining	
Lines 12-14	Combining *RMs*, where each of size *N*, will be performed in *column-wise*. Thus the time complexity of this step is $\sum_{i=1}^{\sqrt{P_G}-1}(ts + tw_{elect} \times i \times \frac{N}{p} \times N)$ that is equal to $ts \times (\sqrt{P_G} - 1) + tw_{elect} \times \frac{P_G - \sqrt{P_G}}{2} \times \frac{N}{p} \times N$. And then the combining will be performed in *row-wise*. Thus, the time complexity of this step is $\sum_{i=1}^{\sqrt{P_G}-1}(ts + tw_{elect} \times i \times \sqrt{P_G} \times \frac{N}{p} \times N)$ that is equal to $ts \times (\sqrt{P_G} - 1) + tw_{elect} \times \frac{P_G - \sqrt{P_G}}{2} \times \sqrt{P_G} \times \frac{N}{p} \times N$.
Second step: Optical data combining	
Lines 15-16	The time complexity of this step, where each *GC* processor will send its *GRM* to G_0 via optical links, is equal to $ts + tw_{optical} \times \frac{N}{p} \times N \times P_G$.
Third step: Main group data combining	
Lines 17-18	All processors in G_0 will send the accumulated *RM* to the *MC* processor. Combining the *RMs* will be performed in *column-wise*. Thus, the time complexity of this step is $\sum_{i=1}^{\sqrt{P_G}-1}(ts + tw_{elect} \times i \times \frac{N}{p} \times N \times P_G + \frac{N}{p} \times N)$ that is equal to $ts \times (\sqrt{P_G} - 1) + tw_{elect} \times \left(\frac{P_G - \sqrt{P_G}}{2}\right) \times \frac{N^2}{p} \times (P_G + 1)$ And then the combining will be performed in *row-wise*. Thus, the time complexity of this step is $\sum_{i=1}^{\sqrt{P_G}-1}(ts + tw_{elect} \times i \times \sqrt{P_G} \times \frac{N}{p} \times N \times P_G + \frac{N}{p} \times N)$ that is equal to $ts \times (\sqrt{P_G} - 1) + tw_{elect} \times \left(\frac{P_G - \sqrt{P_G}}{2}\right) \times \sqrt{P_G} \times \frac{N^2}{p} \times (P_G + 1)$
The overall time complexity of phase 4 is:	

$$
\begin{aligned}
= \Bigg[& \left(ts \times (\sqrt{P_G} - 1) + tw_{elect} \times \frac{P_G - \sqrt{P_G}}{2} \times \frac{N}{p} \times N \right) \\
& + \left(ts \times (\sqrt{P_G} - 1) + tw_{elect} \times \frac{P_G - \sqrt{P_G}}{2} \times \sqrt{P_G} \times \frac{N}{p} \times N \right) \\
& + \left(ts + tw_{optical} \times \frac{N}{p} \times N \times P_G \right) \\
& + \left(ts \times (\sqrt{P_G} - 1) + tw_{elect} \times \left(\frac{P_G - \sqrt{P_G}}{2} \right) \times \frac{N^2}{p} \times (P_G + 1) \right) \\
& + \left(ts \times (\sqrt{P_G} - 1) + tw_{elect} \times \left(\frac{P_G - \sqrt{P_G}}{2} \right) \times \sqrt{P_G} \times \frac{N^2}{p} \times (P_G + 1) \right) \Bigg]
\end{aligned}
$$

$\approx \Theta\left(\frac{N^2 \times \sqrt{P_G}}{2p} \times (p-1)\right) \approx \Theta\left(N^2 \times \sqrt{P_G}\right)$

The overall *parallel run time complexity* of phases 1–4 is shown in Equation (7). Thus, Equation (7) can be reduced to Equation (8).

$$T(N, p) = \Theta(p) + \Theta(N^2 \times \sqrt{P_G}) + \Theta\left(\frac{N}{p} \times N^2\right) + \Theta(N^2 \times \sqrt{P_G}) \quad (7)$$

$$T(N, p) \approx \Theta(p + \frac{N^3}{P} + N^2 \times \sqrt{P_G}) \quad (8)$$

4.2.2 Speedup

The speedup of the PRNN algorithm on OTIS-Mesh is shown in Equation (9).

$$S = \frac{N^3 \times p}{p^2 + N^3 + N^2 \times p \times \sqrt{P_G}} \quad (9)$$

4.2.3 Efficiency

The efficiency of the PRNN algorithm on OTIS-Mesh is shown in Equation (10).

$$E = \frac{N^3}{p^2 + N^3 + N^2 \times p \times \sqrt{P_G}} \quad (10)$$

Table 3 summarizes the parallel run time complexity, speedup and efficiency of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh, where $T_{Distribution}$ is the time needed for distribution phase as a number of communication steps, $T_{Combinng}$ is the time needed for combining phase, and $T_{Parallel}$ is the total parallel run time. As shown in this table, after simplifying the equation of the parallel run time of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh, we notice that the difference between the equations of parallel run time, speedup, and efficiency of OTIS-Hypercube and OTIS-Mesh is the term d in case of OTIS-Hypercube and the term $\sqrt{P_G}$ in case of OTIS-Mesh, where d is the diameter of the Hypercube and $2\sqrt{P_G}$ is the diameter of the Mesh. Note that, $d << 2\sqrt{P_G}$; therefore, OTIS-Hypercube results is superior relative to OTIS-Mesh.

Table 3 Parallel run time complexity, speedup, and efficiency of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh

OTIS-Hypercube	$T_{Distribution}$	$\Theta(N^2 \times d)$, *d is dimension of hypercube and N is number of cities.*
	$T_{Combining}$	$\Theta\left(N^2 - \frac{N^2}{p}\right)$, *N is Distance Matrix (DM)size.*
	$T_{Parallel}$	$\Theta\left(p + \frac{N^3}{P} + N^2 \times d\right)$, *p is number of processors.*
	Speedup	$S = \frac{N^3 \times p}{p^2 + N^3 + N^2 \times p \times d}$
	Efficiency	$E = \frac{N^3}{p^2 + N^3 + N^2 \times p \times d}$
OTIS-Mesh	$T_{Distribution}$	$\Theta(N^2 \times \sqrt{P_G})$
	$T_{Combining}$	$\Theta\left(N^2 \times \sqrt{P_G}\right)$, *N is Distance Matrix (DM) size.*
	$T_{Parallel}$	$\Theta(p + \frac{N^3}{P} + N^2 \times \sqrt{P_G})$
	Speedup	$S = \frac{N^3 \times p}{p^2 + N^3 + N^2 \times p \times \sqrt{P_G}}$
	Efficiency	$E = \frac{N^3}{p^2 + N^3 + N^2 \times p \times \sqrt{P_G}}$

5. Simulation results

A simulation was developed in order to evaluate the performance of PRNN algorithm on both OTIS-Hypercube and OTIS-Mesh optoelectronic architectures. In this section, we present the simulation results obtained from the implementation of the PRNN algorithm on OTIS-Hypercube and OTIS-Mesh optoelectronic architectures.

5.1 Simulation setup

The simulation environment has been set up using java jdk8 under the Eclipse environment. All simulation runs conducted on Intel (R) Core (TM) i7, 3.2 GHz Processor with 16 GB RAM, and 4 MB cache memory and windows 8.1 64-bit as an operating system. To conduct the simulation, we use a startup time equals to 55μs [19], speed of electrical links equals to 250*Mb/s*, and speed of optical links equal to 2.5*Gb/s* [20]. The simulation measures computation time, communication time, speedup and efficiency. Several runs were conducted, where each run was repeated ten times, and the average were considered.

Our implementation has the following classes:

- Distance matrix class: responsible for reading TSPLIB instances.

- Repetitive nearest-neighbour class for performing the sequential nearest-neighbour algorithm.
- Load balancing class for balancing *N* cities among *p* processors.
- OTIS-Hypercube class contains objects of the adopted dimensions of OTIS-Hypercube.
- OTIS-Mesh class contains objects of the adopted dimensions of OTIS-Mesh.

The simulation starts by determining the desired optoelectronic architecture and the dimension chosen by the user. Thus, the number of groups, the number of processors inside each group, and the type of the communication links optical or electrical links will be determined.

Table 4 shows the range of sizes for each optoelectronic architecture, which varies from 16 to 1296 processors. For simplicity, we named each range with class as illustrated in Table 4. These ranges were specified to obtain the desired size for each optoelectronic architecture such that a proper comparison can be accomplished between them. Each row in Table 4 represents a size range, where the values were chosen in a way that minimize the gap of sizes in each range between optoelectronic architectures.

Table 4 Optoelectronic architectures size ranges

Size Range Class	Size Ranges	OTIS-Hypercube	OTIS-Mesh
Class A	16	16	16
Class B	64-81	64	81
Class C	256	256	256
Class D	1024-1296	1024	1296

To conduct the simulation runs, the Travelling Salesman Problem Library (TSPLIB) [17] was used as test set, which enriched the TSP with a great number of sample benchmarks of different TSP types and different formats. Moreover, it contains the current best known solutions for each data instance. Table 5 shows the chosen TSP data instances from both Symmetric and VLSI data sets for this simulation runs. The size of these data instances is varying from 1304 cities to 19289 cities to test the PRNN algorithm on small, medium and large number of cities.

Table 5 TSPLIB data instances [17]

Data Set	Data Instance	Number of Cities	Optimal Solution
TSP Sym	rl1304	1304	252948
TSP Sym	djb2036	2036	6197
VLSI	pcb3038	3038	137694
TSP Sym	bgb4355	4355	12723
VLSI	rl5934	5934	556045
VLSI	xsc6880	6880	21535
VLSI	ida8197	8197	22338
VLSI	rl11849	11849	923473
TSP Sym	xvb13584	13584	37084
VLSI	d15112	15112	1573084
VLSI	frh19289	19289	55798

5.2 Comparative evaluation

In this section, we compare and present detailed discussion of the simulation results of the PRNN algorithm on OTIS-Hypercube and OTIS-Mesh with different granularities ranges from 16 to 1296 processors. The simulation runs were tested using TSPLIB data instances, which are shown in Table 5. Figures from 9-14 demonstrate the performance evaluation metric results obtained from the simulation runs.

Computation time is the time required by each standalone processor to carry out the task. In our algorithm, it is the time in seconds to find the nearest-neighbor route for each city among *N/p* cities. As it should be obvious in Fig. 9. The computation time decreases as the number of processors increases since the term *N/p* will decrease as the number of processors increases. The impact of the number of processors turns out to be more observable in this figure. Thus, the main contribution is in the computation time.

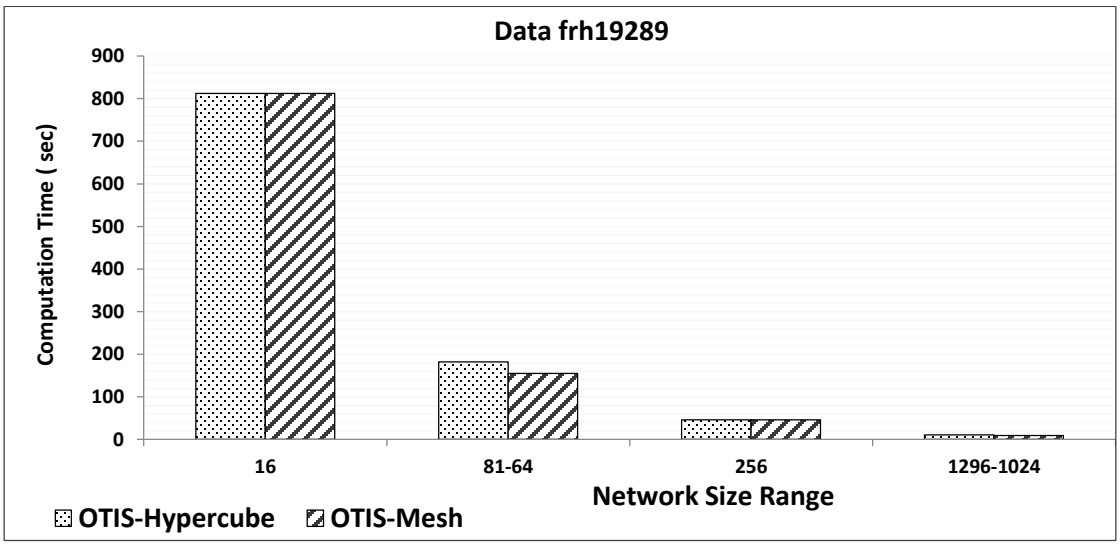

Fig. 9 Computation time of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh for frh19289 TSP instance

PRNN algorithm on OTIS-Hypercube and OTIS-Mesh has the same computation time in class A, since both optoelectronic architectures have the same number of processors. Thus, PRNN algorithm on OTIS-Mesh recorded better computational time in class B than PRNN algorithm on OTIS-Hypercube, since in this class OTIS-Hypercube has less number of processors, which is equal to 64; whereas, OTIS-Mesh has 81 processors. In class C, PRNN algorithm on OTIS-Hypercube and OTIS-Mesh recorded the same computational time since each has 256 processors involved in the computation. Correspondingly, PRNN algorithm on OTIS-Mesh provides slightly better computational time in class D than PRNN algorithm on OTIS-Hypercube, where PRNN algorithm on OTIS-Mesh required 9.8 seconds, while PRNN algorithm on OTIS-Hypercube required 11.1 seconds. Consequently, PRNN algorithm on OTIS-Hypercube gave the worst computational time compared with PRNN algorithm on OTIS-Mesh, since it has less or equal number of processors in each class.

Table 6 Distribution, combining, and total communication time for PRNN algorithm on OTIS-Hypercube and OTIS-Mesh with different size ranges for frh19289 TSP instance

Distribution Time (Seconds)		
Ranges	**OTIS-Mesh**	**OTIS-Hypercube**
16	46.5	46.5
64-81	91.9	69.2
256	137.4	91.9
1024-1296	228.2	114.7
Combining Time (Seconds)		
Ranges	**OTIS-Mesh**	**OTIS-Hypercube**
16	13.1	13.1
64-81	18.6	12.6
256	24.0	11.3
1024-1296	35.0	11.3
Total Communication Time (Seconds)		
Ranges	**OTIS-Mesh**	**OTIS-Hypercube**
16	59.6	59.6
64-81	110.5	81.8
256	161.4	103.2
1024-1296	263.2	126.0

Table 6 demonstrates the distribution time, combining time, and total communication time for the PRNN algorithm on the two optoelectronic architectures with given size ranges for frh19289 TSP instance. As distribution phase relies on distributing the distance matrix with fixed size equal to N^2 among all processors in all groups, the diameter of the optoelectronic architectures played an exclusive role in the number of communication steps for this distribution. Therefore, the PRNN algorithm on OTIS-Hypercube and OTIS-Mesh in class A have the same distribution time since they have the same number of communication steps, which is equal to 5, as shown in Table 7. Note that, PRNN algorithm on OTIS-Hypercube was superior in class B, since 7 communication steps was sufficient for the whole distribution. Conversely, PRNN algorithm on OTIS-Mesh provides higher distribution time with 9 communication steps for the distribution. Although, OTIS-Mesh has higher number of processors in class B, but it recorded the worst distribution time and this proved the argument that the number of processors do not contribute in the distribution phase. The PRNN algorithm on OTIS-Hypercube outperforms PRNN algorithm on OTIS-Mesh in class C. In class D, the PRNN algorithm on OTIS-Mesh recorded the worst distribution time with 21 communication step, while the PRNN algorithm on OTIS-Hypercube recorded the best distribution time with 11 communication steps. It is important to emphasize that only the diameter influenced the distribution results here, since the message size was fixed with this one to all broadcast communication. The optoelectronic architecture with low diameter requires smaller number of communication steps than the one with high diameter. Therefore, the PRNN algorithm on OTIS-Hypercube was predominant in this phase. Interesting observations were seen in this phase, PRNN algorithm on OTIS-Hypercube with 256 processors (Class C) can simulate the communication steps of PRNN algorithm on OTIS-Mesh in class B, because 256 processors of OTIS-Hypercube have the same communication steps as OTIS-Mesh with 81 processors. Accordingly, we can gain higher number of processors and smaller number of communication steps with OTIS-Hypercube in comparison with OTIS-Mesh.

Table 7 Electrical and optical (OTIS) moves for OTIS-Hypercube and OTIS-Mesh

Optoelectronic Architecture	Class A (16)		Class B (64-81)		Class C (256)		Class D (1024-1296)	
	Electrical Move	***OTIS Move***	***Electrical Move***	***OTIS Move***	***Electrical Move***	***OTIS Move***	***Electrical Move***	***OTIS Move***
OTIS-Hypercube	4	1	6	1	8	1	10	1
OTIS-Mesh	4	1	8	1	12	1	20	1

The previous discussion was about the impact of each optoelectronic architecture topological structure on the distribution time. Now, we focus on the discussion about the combining time and its impact factors. As a starting point, it is worth to mention that the communicated message size varies during our combining phase, where each processor in each group starts the gathering phase with different messages of $N \times N/p$ size, then along the process the message size will be enlarged, since each processor in each communication step will concatenate the received data with its own particular message and resend it in the next communication step. Subsequently, each *GC* processor will send message with $N \times N/p \times P_G$ size to its transpose processor in the main group, where P_G is the number of processors inside one group. Each processor inside the main group will starts the main group combining phase with different message of size $N \times N/p + N \times N/p \times P_G$ and this process proceeds until the *MC* processor received the entire data with $N \times N/p \times (p\text{-}1)$ size. Clearly, there are three factors influence the combining phase, which are number of processors, communicated message size $N \times N/p$, and number of communication steps, which depends on the diameter of the optoelectronic architecture.

The combining time in Table 6 exposes the distinctions and the similarities between OTIS-Hypercube and OTIS-Mesh. In this table, the combining time of PRNN algorithm on OTIS-Hypercube decreases as the number of processors increases in each dimension. As a rule of thumb, when the number of processors increases the communicated message size decreases, thus the combining time will decrease too. In addition, the amount of augmentation on the number of communication steps in these optoelectronic architectures is a constant factor of two. So, as outlined previously, the combining time depends on the number of processors. Thus, this increment diminishes this effect and will not have the significant impact when the number of processors increases in each dimension. Therefore, the dominant factor in OTIS-Hypercube will relate to the increment of the number of processors. While combining time of PRNN algorithm on OTIS-Mesh increases with the increment on the number of processors; this is due to the large difference between the numbers of communication steps from one dimension to another in this optoelectronic architecture. So, with the augmentation in number of processors, this difference will influence the behaviour of the combining time. In general, the diameter becomes worse as OTIS-Mesh size increases and this leave a substantial impact on the number of communication steps in this optoelectronic architecture.

Fig. 10 clarifies the total communication time for PRNN algorithm on OTIS-Hypercube and OTIS-Mesh, where PRNN algorithm on OTIS-Hypercube recorded the best communication time and PRNN algorithm on OTIS-Mesh recorded the worst communication time. An interesting observation of PRNN algorithm on OTIS-Hypercube with 1024 processors in class D, where it can achieve near the communication time of PRNN algorithm on OTIS-Mesh with 81 processors in class B. Thus, we can gain less communication time and higher number of processors on OTIS-Hypercube in comparison with OTIS-Mesh.

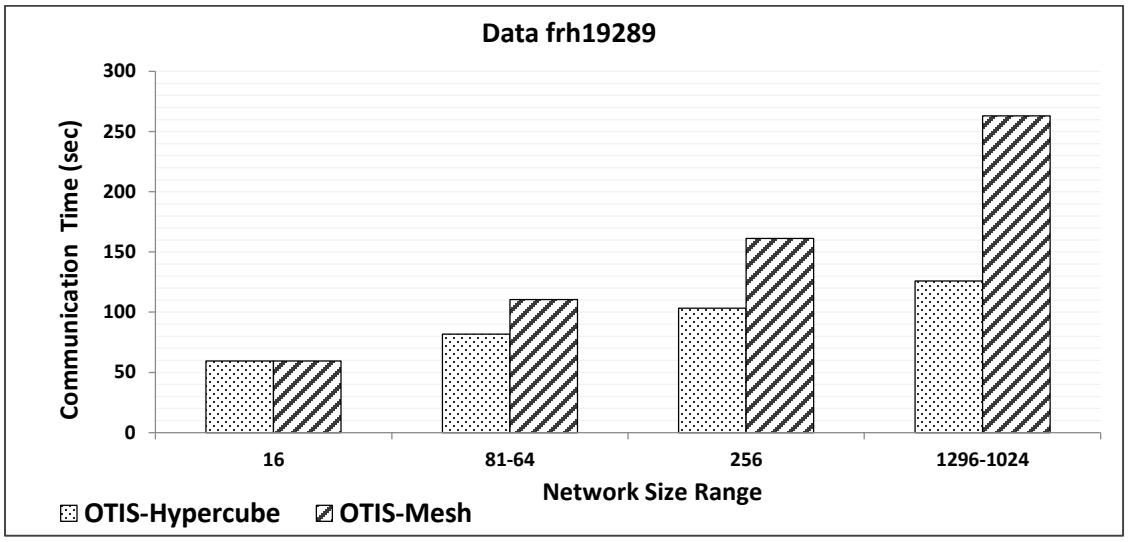

Fig. 10 Communication time of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh for frh19289 TSP instance

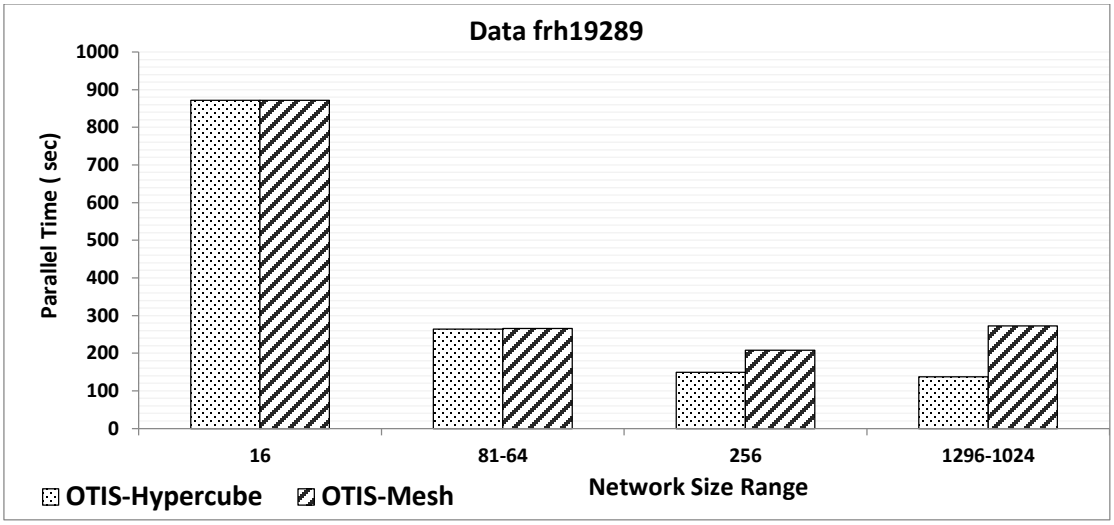

Fig. 11 Parallel time of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh for frh19289 TSP instance

Parallel time is the time for the whole parallel process, it elapses from starting of the distribution process till finishing the final computation by the *MC* processor. Fig. 11 reflects the results regarding the parallel time, which was calculated based on the summation of distribution time, computation time, communication time and the time required by *MC* processor to find optimal route. It is obvious from this figure, PRNN algorithm on OTIS-Hypercube recorded the smallest parallel time and PRNN algorithm on OTIS-Mesh recorded the largest parallel time. A careful look at this figure shows in class A both optoelectronic architectures have similar parallel time since both have the same structure and same number of processors (16 processors), as depicted in Fig. 1. Likewise, they recorded the same parallel time in class B, since the loss in

communication time compensated the increment in the computation time as illustrated in the previous discussion. In classes C and D, both manifested the differences between the optoelectronic architectures, where PRNN algorithm on OTIS-Hypercube acquired less parallel time than PRNN algorithm on OTIS-Mesh.

Speedup results were illustrated in Fig. 12. Generally, this figure shows the growing of speedup as the number of processors increases, this growth can almost approach the number of processor. In particular, in classes A and B, speedup approaches to 14.3 and 47.4, respectively, for PRNN algorithm on OTIS-Hypercube. This is due to the large ratio between the required time for the computation operation over the required time for the communication operation. However, this ratio gets smaller in classes C and D, owing to the increment on the number of processors while the data size remains small relative to these processors, so that the computation time becomes small relative to the communication time. This result will definitely be better with larger data instances.

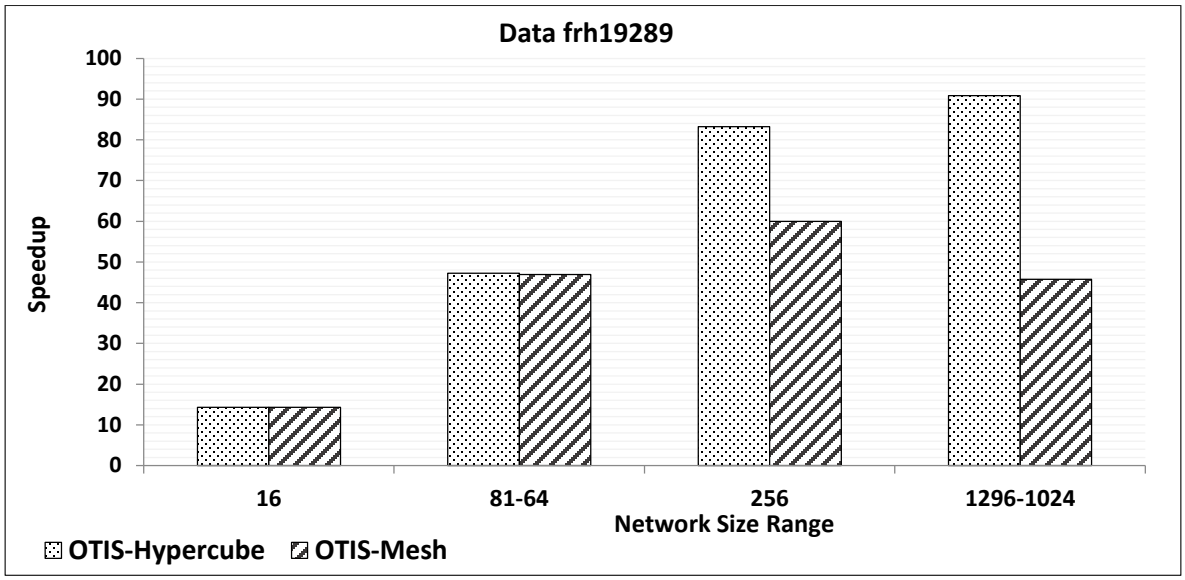

Fig. 12 Speedup of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh for frh19289 TSP instance

In the same way, a careful look at PRNN algorithm on OTIS-Mesh reveals a slight degradation particularly in class D, owing to the fact that class D for OTIS-Mesh contains large number of processors, which is equal to 1296 processor. Thus, the computation time for each processor in class D becomes smaller and simultaneously the communication time becomes large proportional to the communication time in class C, this explains why this degradation occurred.

The previous discussion was about the whole figure which showed in general the influence of the number of processors on the speedup. Now, let's focus our discussion on each class separately. Beginning with class A, you can observe that the speedup in class A for PRNN algorithm on OTIS-Hypercube and OTIS-Mesh is the same, since they have same topological structure, such as number of processors, number of groups, the way these processors are connected in the groups and the diameter. Therefore, they can achieve the same speedup. Intuitively, classes C and D clarified the difference between the optoelectronic architectures in speedup, as we can see, PRNN algorithm on OTIS-Hypercube outperforms PRNN algorithm on OTIS-Mesh in speedup, due to its factor network (hypercube) which fulfilled less communication steps than mesh, because of its low diameter.

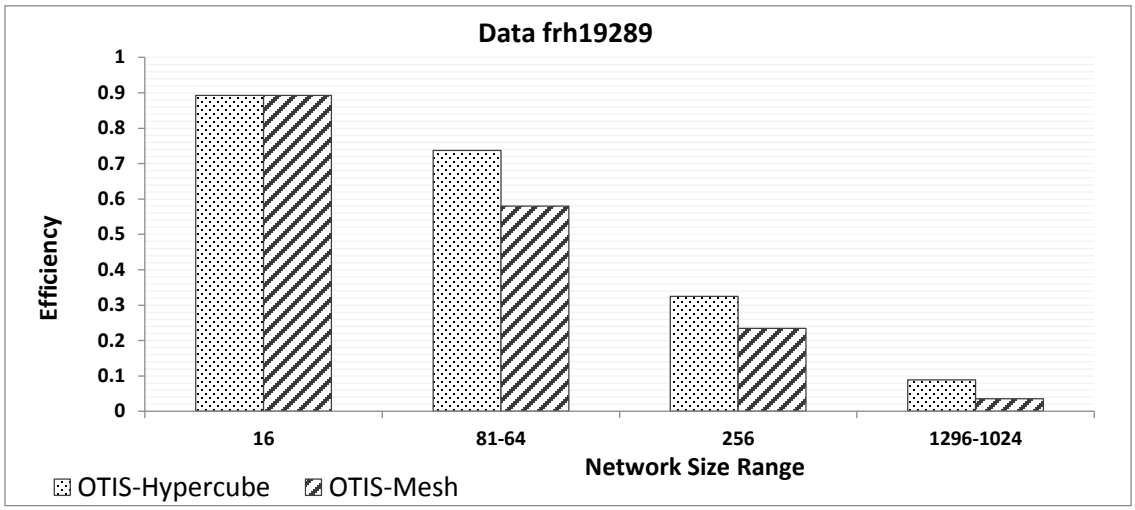

Fig. 13 Efficiency of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh for frh19289 TSP instance

Fig. 13 depicts the efficiency for PRNN algorithm on OTIS-Hypercube and OTIS-Mesh. An intuitive result is the decreasing of efficiency as the number of processors increases, since efficiency is defined as the ratio between speedup and the number of processors. So, when the number of processors increases, the ratio will decrease. A careful examination of class A in this figure denotes the achieved excellent efficiency, which approaches to 0.9 for PRNN algorithm on OTIS-Hypercube and OTIS-Mesh. This equivalence of efficiency between these optoelectronic architectures considers the mentioned reasons of the similarity between the topological structures of these two optoelectronic architectures. In general, the PRNN algorithm on OTIS-Hypercube outperforms the PRNN algorithm on OTIS-Mesh in regard to efficiency for classes B, C and D.

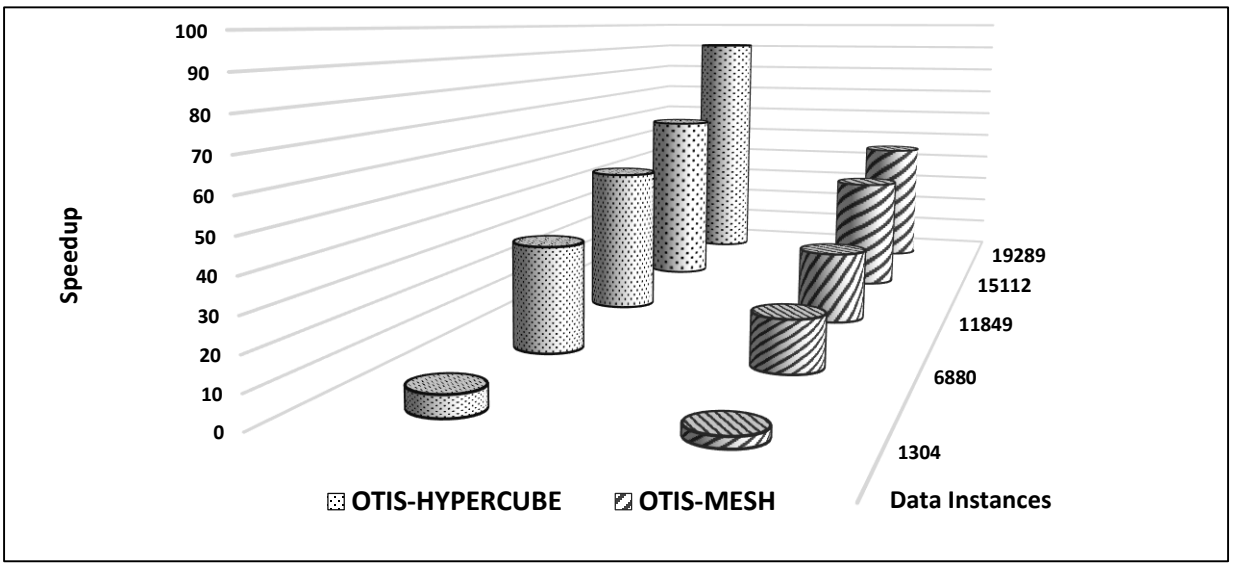

Fig. 14 Speedup of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh for class D of various TSP instances

Fig. 14 demonstrates the speedup of five different TSP instances tested on OTIS-Hypercube and OTIS-Mesh in class D. They are rl1304, xsc6880, rl11849, d15112 and frh19289 TSP instances [17]. The name of the TSP instance indicates the number of cities, for example the data instance xsc6880 has 6880 cities, as shown in Table 5. Fig. 14 shows increasing of speedup when increasing of TSP instance size for both optoelectronic architectures. It is important to mention that we chose this size range despite the low speedup it provided compared to other size ranges as justified before, because it can show clearly the differences between OTIS-Hypercube and OTIS-Mesh in regard to speedup. As shown in the figure, PRNN algorithm on OTIS-Hypercube gained the best speedup among all data instances.

Solution quality of a heuristic algorithm is a major concern since it determines how close the produced solution to the optimal one. Consequently, since the optimal solution of the chosen TSP instances is known, then we are able to measure the solutions quality based on them using the percentage deviation [21] as illustrated in Equation (11), where S_q, F_S, O_S denote *Solution quality*, *Found Solution*, and *Optimal Solution*, respectively.

$$S_q = \frac{F_S - O_S}{O_S} \times 100\% \quad (11)$$

The PRNN algorithm recorded better solutions than the sequential nearest-neighbor algorithm. The starting city in the nearest-neighbor algorithm can play an important role in the solution quality. So, we decided to apply the algorithm *N* times, where *N* is number of cities and each time with different starting city, then choose the route with the minimum distance. This helped to obtain better results than applying the sequential nearest-neighbor algorithm on a random starting city. Thus, in Table 8, you can notice that the algorithm gave better solutions, for example, the percentage deviation of the best solutions gave an average of 20.2% approximate solution within the optimal solution. The percentage deviation of the average solutions gave an average of 25.2% approximate solution within the optimal solution. Applying any random initial city, gave in the worst-case an average of 31.2% approximate solutions within the optimal solution. With respect to the solution quality, the majority of heuristic algorithms solution quality become poor as increasing of the data size [22]. This can be clearly seen in the first three rows of this table compared to the rest of it, where the best percentage deviation gave an average of 15.4 % approximate solution within the optimal solution. On the other hand, it is important to mention that the solution quality was not influenced neither by the type of optoelectronic architecture nor by the number of processors; therefore, the obtained solutions were similar on both optoelectronic architectures for all size ranges.

Table 8 Tour quality solutions; Δ stands for percentage deviation from the optimal solution

TSP Instance	Optimal solution	Best solution	Average solution	Worst solution	Best Δ %	Average Δ %	Worst Δ %	Starting City
wi29	27603	32164	34974	38042	16.5	26.7	37.8	7
eil76	538	608	665	710	13.0	23.6	32.0	52
ch130	6110	7129	7735	8837	16.7	26.6	44.6	3
rl1304	252948	306195	322047.7	339653	21.1	27.3	34.3	901
djb2036	6197	7645	7899.4	8274	23.4	27.5	33.5	552
pcb3038	137694	169009	173377.9	179015	22.7	25.9	30.0	2198
bgb4355	12723	15623	16052.1	16612	22.8	26.2	30.6	2231
rl5934	556045	657056	676459	702041	18.2	21.7	26.3	5417
xsc6880	21535	26243	26948.6	27963	21.9	25.1	29.8	1056
ida8197	22338	27513	28278.9	28964	23.2	26.6	29.7	5473
rl11849	923473	1100013	1126208.9	1147853	19.1	22.0	24.3	5849
d15112	1573084	1910419	1944711.3	1982389	21.4	23.6	26.0	9554
frh19289	55798	68360	69604.6	70930	22.5	24.7	27.1	15456

6. Conclusions and future work

In summary, this paper introduced and evaluated PRNN algorithm for solving TSP on two selected OTIS optoelectronic architectures namely: OTIS-Hypercube and OTIS-Mesh. We discussed in details four phases of the algorithm, these phases are load balancing phase, data distribution phase, sequential nearest-neighbor algorithm phase and data combining phase. Each phase was evaluated analytically and was carried out by simulation on each optoelectronic architecture. The conducted runs examined the algorithms over different data instances from TSPLIB with various sizes. For comparison purposes, we suggested four classes of size ranges such that a performance evaluation of these optoelectronic architectures can be established. We evaluated the performance in terms of number of communication steps, parallel run time, speedup, and efficiency. The simulation results achieved high speedup among the two optoelectronic architectures of interest. It was clear from the simulation and the analytical results that OTIS-Hypercube gained the best results in all the performance metrics in comparison to OTIS-Mesh. The results were justified based on the factors that influenced by both the computation and communication time; such as the diameter of the factor network exists in each group, the number of communication steps, the communicated message size, and the number of processors in each optoelectronic architecture.

These optoelectronic architectures share attractive features; since partitioning OTIS optoelectronic architecture into N groups of N processors can support large scale systems with less cost and less complexity design. These interesting features enabled us to adopt the algorithm in a way that meet the purpose to obtain high quality solution in less time. Moreover, they helped to record near linear speedup approaches to 14.3 and high efficiency approaches to 0.9 in case of having 16 processors. Therefore, this will stimulate researchers to apply other problems in computer science field as a future work. Also, a comparative study can be applied between each OTIS optoelectronic architecture with its factor network; for example, OTIS-Hypercube and Hypercube to study the amount of gained performance between them.

References

1. Marsden G, Marchand P, Harvey P, Esener S (1993) Optical transpose interconnection system architectures. Optics Letters 18(13):1083-1085
2. Rajasekaran S, Reif J (2008) Handbook of parallel computing Models, Algorithms and Applications, CRC Press, USA
3. Lucas KT, Jana PK (2010) Parallel algorithms for finding polynomial Roots on OTIS-torus. Journal of Supercomputing 54(2): 139-153
4. Jana P, Mallick D (2010) OTIS-MOT: an efficient interconnection network for parallel processing. Journal of Supercomputing 59(2): 920-940
5. Mahafzah B, Sleit, Hamad N, Ahmad E, Abu-Kabeer T (2012) The OTIS hyper hexa-cell optoelectronic architecture. Computing 94(5):411-432
6. Wang C-F, Sahni S (1998) Basic operations on the OTIS-mesh optoelectronic computer IEEE Trans. Parallel Distributed Systems 9(12):1226-1236
7. Osterloh A (2000) Sorting on the OTIS-mesh. In: Proceedings of the 14th International Parallel and Distributed Processing Symposium (IPDPS'00), pp 269-74
8. Mahafzah B, Tahboub R, Tahboub O (2010) Performance evaluation of broadcast and global combine operations in all-port wormhole-routed OTIS-mesh interconnection networks. Cluster Computing 13(1): 87-110
9. Mahafzah B, Jaradat B (2008) The load balancing problem in OTIS-hypercube interconnection networks. Journal of Supercomputing 46(3): 276-97
10. Deb S, Fong S, Tian Z, Wong RK, Mohammed S, Fiaidhi J (2016) Finding approximate solutions of NP-hard optimization and TSP problems using elephant search algorithm. Journal of Supercomputing 72(10):3960-3992
11. Matai R, Singh SP, Mittal ML (2010) Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches. In: Traveling Salesman Problem, Theory and Applications. pp 1– 24
12. Cormen T, Leiserson, C, Rivest R, Stein C (2001) Introduction to Algorithms. MIT press, London
13. Kang S, Kim S-S, Won J, Kang Y-M (2016) GPU-based parallel genetic approach to large-scale travelling salesman problem. Journal of Supercomputing 72(11):4399-4414
14. Marinakis Y (2009) Heuristic and Metaheuristic Algorithms for the Traveling Salesman Problem. In: Floudas, CA, Pardalos, PM (eds) Encyclopedia of Optimization, Springer, USA pp 1498–1506
15. Reinelt G (1994) The Traveling Salesman: Computational Solutions for TSP Applications. Lecture Notes in Computer Science 840:73-97
16. Zane F, Marchand P, Paturi R, Esener S (2000) Scalable network architectures using the optical transpose interconnection system (OTIS). Journal of Parallel and Distributed Computing 60(5):521-538
17. Reinelt G (1991) TSPLIB: A Traveling Salesman Problem Library. ORSA Journal on Computing. 3(4):376-384
18. Grama, A, Gupta A, Karyp G, Kumar G (2003) Introduction to Parallel Computing. Addison Wesley, USA
19. Hennessy JL, Patterson DA (2011) Computer architecture: A Quantitative Approach. Morgan Kaufmann

20. Kibar O, Marchand P, Esener S (1998) High speed CMOS switch designs for free-space optoelectronic MINs. IEEE Trans Very Large Scale Integr (VLSI) Syst 6(3):372–386
21. Ansari AQ, Katiyar S (2015) Comparison and analysis of solving travelling salesman problem using GA, ACO and hybrid of ACO with GA and CS. In: Computational Intelligence: Theories, Applications and Future Directions (WCI), 2015 IEEE Workshop pp 1-5
22. Johnson DS, Aragon CR, McGeoch LA, Schevon C (1989) Optimization by Simulated Annealing: An Experimental Evaluation: Part I, Graph Partitioning. Operations Research, 37(6):865–892