J Supercomput (2018) 74:1–36
https://doi.org/10.1007/s11227-017-2102-y

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

Aryaf Al-Adwan[1] · **Basel A. Mahafzah**[1] · **Ahmad Sharieh**[1]

Published online: 6 July 2017

Abstract Over the past years, researchers drew their attention to propose optoelectronic architectures, including optical transpose interconnection system (OTIS) networks. On the other hand, there are limited attempts devoted to design parallel algorithms for 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, a parallel repetitive nearest neighbor algorithm for solving the symmetric traveling salesman problem on OTIS-Hypercube and OTIS-Mesh optoelectronic architectures is presented. This algorithm has been evaluated analytically and by simulation on both optoelectronic architectures in terms of number of communication steps, parallel run time, speedup, efficiency, cost and communication cost. The simulation results attained almost near-linear speedup and high efficiency among the two selected optoelectronic architectures, where OTIS-Hypercube gained better results in comparison with OTIS-Mesh.

Keywords Parallel heuristics algorithm · Nearest neighbor Algorithm · Traveling 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

✉ Basel A. Mahafzah
b.mahafzah@ju.edu.jo

[1] Computer Science Department, King Abdullah II School for Information Technology, The University of Jordan, Amman 11942, Jordan

these processing elements play a 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 traveling 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]. This problem can be solved using exact algorithms that evaluate all possible solutions to provide the optimal one for a 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 get a suboptimal solution rather than the optimal one. One of those were the constructive methods that build a tour then stop when one solution is obtained, such as nearest neighbor heuristic algorithm [14,15]. One variation of this algorithm is the repetitive nearest neighbor algorithm, where the minimum route is obtained by applying the nearest neighbor algorithm on each city as a starting city, then finding the minimum route among all the computed ones. However, it is considered inefficient in terms of time, since this involves iterative processes that exhaust the available resources, particularly when it is applied to a huge input size using a sequential machine. Thus, TSP should be solved by parallel algorithms that can perform the computations of large instances of cities in less time. Consequently, in order to solve TSP using the attractive features of optoelectronic architectures and to carry out a performance evaluation between OTIS-Hypercube and OTIS-Mesh, this paper presents a parallel repetitive nearest neighbor (PRNN) algorithm for solving TSP on OTIS-Hypercube and OTIS-Mesh optoelectronic architectures. The algorithm is evaluated analytically under the following performance metrics: number of communication steps, parallel run time, speedup, efficiency, cost and communication cost on OTIS-Hypercube and OTIS-Mesh. Also, its performance was evaluated by simulation runs on both optoelectronic architectures. To the best of our knowledge, there

is no work that has been investigated the performance of the OTIS optoelectronic architecture in solving TSP.

The organization of the paper is as follows: Sect. 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 an analytical evaluation of this algorithm on both optoelectronic architectures. Section 5 shows the simulation setup and results. Finally, Sect. 6 summarizes the overall work and presents some suggested future work.

2 OTIS optoelectronic architecture

OTIS optoelectronic architecture provides efficient connections with high scalability and lower complexity among the connected processors, in addition to utilizing the optical technology in sending and receiving data between the connected groups [1]. They are all (group)-to-all (group) network with P processors. This network consists of groups, where these groups are connected as a basic network: such as ring, mesh, hypercube, etc., or these 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 processors in different groups is done by transposing processor's and group's labels. For example, the third processor in the second group is connected by an 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 or P processors that are partitioned into n groups, and each group contains n or P_G processors interconnected as a hypercube of dimension d, which is equal to $\log n$. For example, in 16-processor 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].

2.2 OTIS-Mesh

OTIS-Mesh is another example of OTIS basic networks, which consists of n^2 or P processors that are partitioned into n groups, and each group contains n or P_G processors interconnected as a two-dimensional $\sqrt{n} \times \sqrt{n}$ mesh. The communication

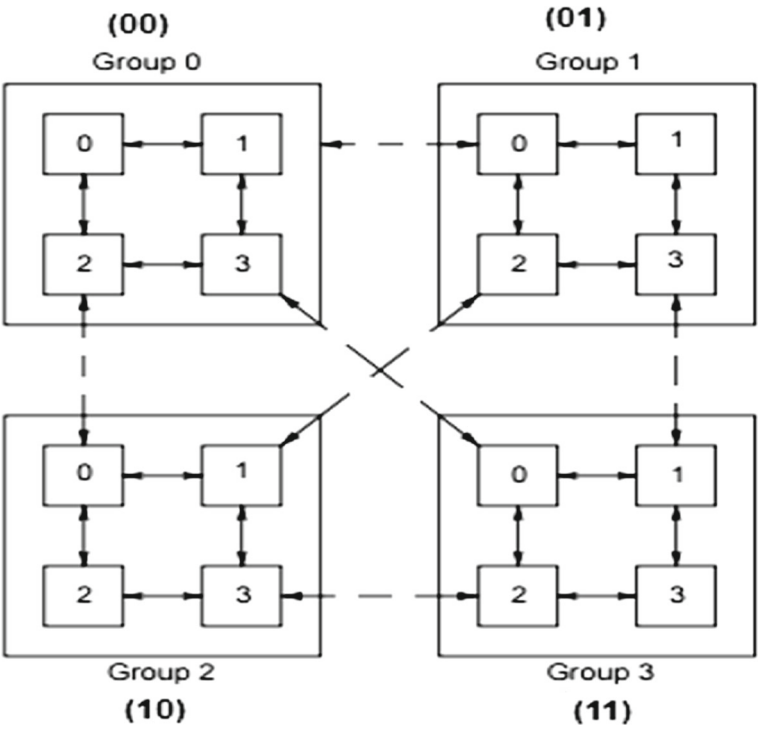

Fig. 1 Two-dimensional OTIS-Hypercube [2]

links within each group are electronic and between groups are optical. For example, in 81-processor 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 shown 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.

3 Solving TSP using PRNN algorithm

Tour construction algorithms can be identified by building a solution through a sequence of steps based on immediate advantageous. These steps keep evolving 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 Table 1.

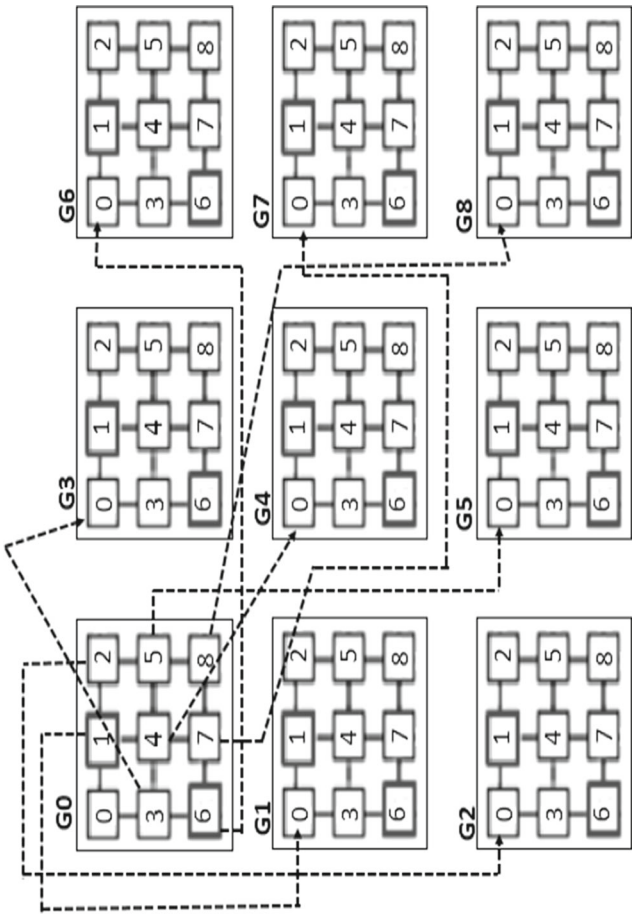

Fig. 2 OTIS-Mesh of nine groups and each group is 3×3 two-dimensional mesh

Table 1 Sequential nearest neighbor algorithm [14]

Step 1: pick a random city as a current vertex Cv
Step 2: find the nearest unvisited city Nu
Step 3: set the current vertex Cv as Nu
Step 4: repeat Step 2 until all cities are visited
Step 5: link the starting city with the last one to form the tour

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 the 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 a 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 medium 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, in a simple way without compromising the solution quality and with less communication time.

In this section, a well-known TSP heuristic algorithm is selected to be parallelized over interconnections of interest. The parallel repetitive nearest neighbor (PRNN) algorithm along with its illustration followed by its analytical evaluation is presented first.

3.1 PRNN algorithm on OTIS-Hypercube

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 a number of cities that is not a power of two [17]. Consequently, we have introduced a load balancing algorithm, as illustrated in Fig. 3, and the PRNN algorithm on OTIS-Hypercube is illustrated in Fig. 4.

3.1.1 Phase 1: load balancing phase

The load balancing phase is achieved by the *MC* processor via applying the load balancing algorithm (Fig. 3), which guarantees only one extra city for the balanced

```
Algorithm1: Load balancing algorithm
Input: Number of Cities N, Number of Processors P
Output: Number of Cities N is load balanced among Processors P
Main Coordinator (MC) performs the following:
1.    M = N / P;
2.    extraCities = P× (M mod 1);
3.    for all processors P
4.        do
5.            Assign (N_over_P) cities for each processor;
6.    if (M 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. 3 Load balancing algorithm

```
Algorithm2: 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. 3)
   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 ACA internally utilizing the hamming distance through the electronic 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 electronic links;
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 G0;
18. MC processor combines the collected group route matrices and finds the minimum route and its cost;
```

Fig. 4 PRNN algorithm on OTIS-Hypercube

processors. This clearly can be demonstrated through lines 1–13 in Fig. 3. 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

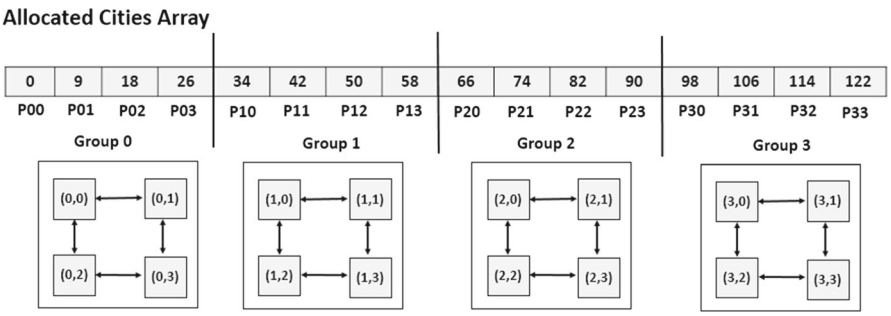

Fig. 5 *ACA* created by *MC* processor

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 array (*ACA*) 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. 4.

The PRNN algorithm on OTIS-Hypercube in Fig. 4 is illustrated in more details as in Sects. 3.1.2–3.1.4.

3.1.2 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 to 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. 4): *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. 3. According to this algorithm, each processor will obtain a set of cities, 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 ACA* are located at *MC* processor,

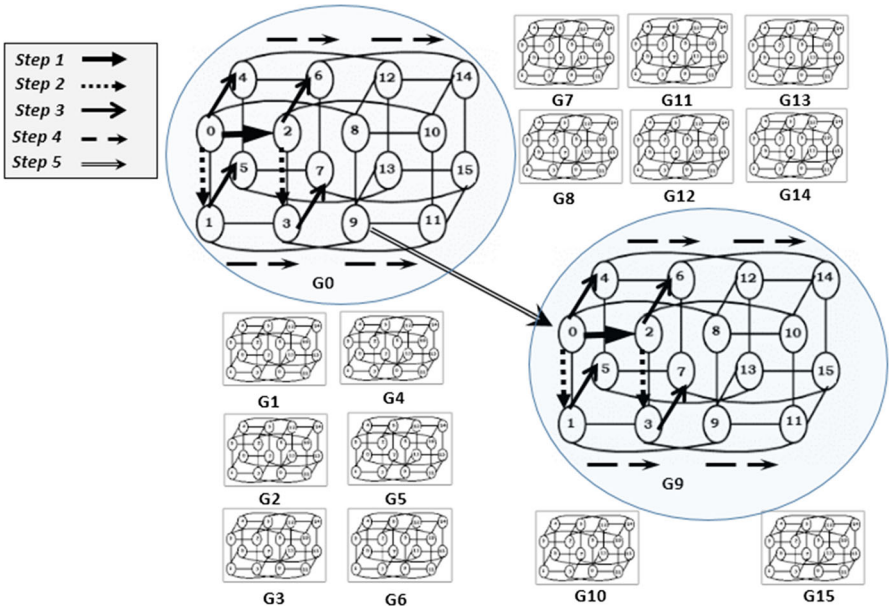

Fig. 6 Distribution of distance matrix (*DM*) on OTIS-Hypercube

then it is responsible to send a copy of the *DM* and *ACA* to all other processors using one to all broadcast. Initially, *MC* processor has *DM* and *ACA*, and at the termination of this broadcasting there will be P_G copies of *DM and ACA*, 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. 6, through steps 1, 2, 3 and 4. This will take log P_G electronic moves. Then, all the processors that receive *DM and ACA* in *MG* start the optical distribution of data as shown in the next step.

II. *Optical distribution of data* (lines 6–7 in Fig. 4): All processors in G_0 (in parallel) start sending *DM and ACA* through the optical links to their corresponding processors in other groups. This will require 1 OTIS move. For example, in Fig. 6 (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. 4): 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 ACA* to each processor in its group using the hamming distance, as shown in Fig. 6, through steps 1, 2, 3 and 4. Again, this will take $log\ P_G$ electronic moves.

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

3.1.3 Phase 3: local repetitive nearest neighbor phase

I. After finishing phase 2, all processors in OTIS-Hypercube will have the distance matrix and the allocated cities array. The allocated cities array determines for each processor the starting city in order to apply the sequential nearest neighbor on; for example, processor 2 in group 0 will apply the sequential repetitive nearest neighbor on cities 18 through 25, as depicted in Fig. 5. Such that P02 will apply the sequential nearest neighbor at the first time on city 18 and stores the route in its route matrix (RM), then applies the sequential nearest neighbor on city 19 and stores the route in its RM and so on. Therefore, in parallel, all processors in OTIS-Hypercube 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 RM such that each processor will have its own *RM*. Thus, phase 3 is shown in Fig. 4, lines 10–11.

3.1.4 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. 4). 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 a 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. 4). 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. 4). Each processor will combine the collected *GRM* with its own *RM* and sends it to its neighbor that differs in dth 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. 7. Both load

Algorithm3: 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. 3)
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 *ACA using one-to-all broadcast to processors in its corresponding column through the electronic 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 ($\sqrt{P_G}$-1)*
*14. **do** in parallel*
15. Each processor routes its route matrix (RM) to processors in its corresponding column through the electronic 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. 7 PRNN algorithm on OTIS-Mesh

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.

3.2.1 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. 7): *MC* processor must send a copy of the *DM* and *ACA* 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 electronic links. At the end of this phase, P_G copies of *DM* and *ACA* will be sent to P_G processors in G_0, and this requires $2\sqrt{P_G}$-1 communication steps. Then, all the processors that received *DM* and *ACA* in *MG* start the optical distribution of data as shown in the next step.

II. *Optical distribution of data* (lines 7–8 in Fig. 7): All processors in G_0 (in parallel) start sending *DM* and *ACA* 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. 7): 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 *ACA* to each processor in its group through *row-wise* and *column-wise* phases, as have been done in electronic main group distribution.

3.2.2 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. 7). 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 send 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. 7). 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. 7). 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 used to evaluate the algorithm are parallel time complexity, speedup, efficiency, cost and communication cost.

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, efficiency, cost and communication cost.

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 Eq. (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.*

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

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. 4, as shown in Table 2.

The overall *parallel run time complexity* of phases 1–4 is shown in Eq. (2).

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

Equation (2) can be reduced to Eq. (3).

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

4.1.2 Speedup

The speedup (S) 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

Table 2 Run time complexity

Phase 1 (*Load balancing phase*)	
Line 1	*MC* processor in G_0 executes the load balancing algorithm in Fig. 3. 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
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 *ACA* 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 *ACA* 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 *ACA* 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: $= (d \times t_s + tw_{elect} \times (N^2 + P) \times d) + (t_s + tw_{optical} \times (N^2 + P)) + (d \times t_s + tw_{elect} \times (N^2 + P) \times d) \approx \Theta\left(N^2 \times d\right)$	
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
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 $\sum_{i=1}^{log P_G} (ts + tw_{elect} \times 2^{i-1} \times \frac{N}{P} \times N)$ that is equal to $ts \times log P_G + tw_{elect} \times (P_G - 1) \times \frac{N}{P} \times N$

Table 2 continued

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 $\sum_{i=1}^{log P_G} (ts+tw_{elect} \times 2^{i-1} \times \frac{N}{P} \times N \times P_G + \frac{N}{P} \times N)$ that is equal to $ts \times log P_G + tw_{elect} \times (P_G - 1) \times \frac{N^2}{P} \times (P_G + 1)$
The overall time complexity of phase 4 is: $= (ts \times log P_G + tw_{elect} \times (P_G - 1) \times \frac{N}{P} \times N) + (ts + tw_{optical} \times \frac{N}{P} \times N \times P_G) + (ts \times log P_G + tw_{elect} \times (P_G - 1) \times \frac{N^2}{P} \times (P_G + 1)) \approx \Theta\left(N^2 - \frac{N^2}{P}\right)$, where P is equal to P_G^2	

to solve the same problem on parallel machine, that is, $S = T_S / T_P$ [18], where T_S is the time required by the sequential algorithm and T_P is the time required by the parallel algorithm. The sequential version of repetitive nearest neighbor requires N^3 time complexity, and PRNN algorithm requires time complexity which is illustrated in Eq. (3). Therefore, the speedup of the PRNN algorithm on OTIS-Hypercube is shown as in Eq. (4).

$$S = \frac{N^3 \times P}{P^2 + N^3 + N^2 \times d \times P} \tag{4}$$

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 the number of processors. Therefore, the efficiency of the PRNN algorithm on OTIS-Hypercube is shown in Eq. (5).

$$E = \frac{N^3}{P^2 + N^3 + N^2 \times d \times P} \tag{5}$$

4.1.4 Cost

The cost of solving any problem on a parallel machine can be defined as the total time required by all the processors in the parallel machine to solve the problem. It can be calculated by multiplying the number of processors with the parallel time, where $cost = P \times T_P$ [18]. T_P is presented in Eq. (3), and P is the number of processors in

the optoelectronic architecture. Therefore, the cost of the PRNN algorithm on OTIS-Hypercube is shown in Eq. (6).

$$Cost = P^2 + N^3 + N^2 \times d \times P \tag{6}$$

4.1.5 Communication cost

Communication cost presents the total number of parallel network links which are utilized during solving any problem. In our case, it is the total number of electronic links and optical links used by PRNN algorithm on OTIS-Hypercube. The communication cost of PRRN algorithm depends on the proposed communication pattern among both the distribution and combining phases. As depicted in Table 3, electronic main group distribution in OTIS-Hypercube requires $P_G - 1$ electronic links, due to the one to all broadcast communication pattern. For instance, in the case of 64 processors in OTIS-Hypercube, the one to all broadcast operation is performed among three steps: in the first step, the number of utilized electronic links is 1. In the second step, the number of utilized links is 2, since two processors will send the received distance matrix in parallel. In the third step, the number of utilized links is 4, since 4 processors will send the received distance matrix in parallel, so the total number of the utilized electronic links in this phase is 7. The optical distribution of data requires $P_G - 1$ optical links, which is equal to the number of groups in OTIS-Hypercube minus one. The reason of subtracting one from the number of groups is related to the topological structure of the OTIS networks, where all processors in group 0 except processor 0 are connected via optical links to their transposed processors in other groups. Now, in inter-group distribution of data, the number of the utilized electronic links is equal to $P_G^2 - 2P_G + 1$, which is obtained by multiplying the number of groups minus one and the number of electronic links that is utilized during the distribution inside one group. Note that, the communication cost for the combining phase is like the communication cost for the distribution phase but in reverse order. Therefore, the total communication cost of applying PRNN algorithm on OTIS-Hypercube is illustrated in Eq. (7). Consequently, the communication cost of OTIS-Hypercube depends directly on the number of processors in each group.

$$Comm_{Cost} = 2(P_G^2 - 1) \tag{7}$$

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*.

Table 3 Communication cost of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh

Phase	Communication phases	OTIS-Hypercube communication cost	OTIS-Mesh communication cost
Distribution	Electronic main group distribution	$P_G - 1$	$\left(\sqrt{P_G} - 1\right) + \left(P_G - \sqrt{P_G}\right)$
	Optical distribution of data	$P_G - 1$	$P_G - 1$
	Inter-group distribution of data	$P_G^2 - 2P_G + 1$	$P_G^2 - 2P_G + 1$
Combining	Inter-group data combining	$P_G^2 - 2P_G + 1$	$P_G^2 - 2P_G + 1$
	Optical data combining	$P_G - 1$	$P_G - 1$
	Main group data combining	$P_G - 1$	$\left(\sqrt{P_G} - 1\right) + \left(P_G - \sqrt{P_G}\right)$

Theorem 2 *The average-case time complexity of PRNN algorithm on OTIS-Mesh is shown in Eq. (8), 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.*

$$T(N, p) = \Theta\left(P + \frac{N^3}{P} + N^2 \times \sqrt{P_G}\right) \tag{8}$$

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. 7, as shown in Table 4. □

The overall *parallel run time complexity* of phases 1–4 is shown in Eq. (9).

$$T(N, p) = \Theta(P) + \Theta(N^2 \times \sqrt{P_G}) + \Theta\left(\frac{N}{P} \times N^2\right) + \Theta(N^2 \times \sqrt{P_G}) \tag{9}$$

Equation (9) can be reduced to Eq. (10).

$$T(N, p) \approx \Theta(P + \frac{N^3}{P} + N^2 \times \sqrt{P_G}) \tag{10}$$

4.2.2 Speedup

The speedup (S) is equal to T_S / T_P, where T_S is the time required by the sequential algorithm to perform any task on sequential machine and T_P is the time required by the parallel algorithm to perform the same task on a parallel machine. The sequential version of repetitive nearest neighbor requires N^3 time complexity, and PRNN algorithm requires time complexity which is illustrated in Eq. (10). Thus, the speedup of the PRNN algorithm on OTIS-Mesh is shown in Eq. (11).

$$S = \frac{N^3 \times P}{P^2 + N^3 + N^2 \times P \times \sqrt{P_G}} \tag{11}$$

Table 4 Run time complexity

Phase 1 (*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
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 *ACA* 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 *ACA* 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: $(2 \times (\sqrt{P_G} - 1) \times t_s + tw_{elect} \times (N^2 + P) \times 2 \times (\sqrt{P_G} - 1)) + (t_s + tw_{optical} \times (N^2 + P)) + (2 \times (\sqrt{P_G} - 1) \times t_s + tw_{elect} \times (N^2 + P) \times 2 \times (\sqrt{P_G} - 1)) \approx \Theta(N^2 \times \sqrt{P_G})$	
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 4 continued

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:

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

4.2.3 Efficiency

The efficiency (E) performance metric can be used to measure how much the processors being utilized [18] in the optoelectronic architecture. It is the ratio between speedup and the number of processors. Therefore, the efficiency of the PRNN algorithm on OTIS-Mesh is shown in Eq. (12).

$$E = \frac{N^3}{P^2 + N^3 + N^2 \times P \times \sqrt{P_G}} \tag{12}$$

4.2.4 Cost

The cost of solving any problem on any parallel machine can be calculated by multiplying the number of processors in this parallel machine with the parallel time needed to solve the problem, where $cost = P \times T_P$ [18]. T_P is presented in Eq. (10), and P is the number of processors in the optoelectronic architecture. Therefore, the cost of the PRNN algorithm on OTIS-Mesh is shown in Eq. (13).

$$\text{Cost} = P^2 + N^3 + N^2 \times \sqrt{P_G} \times P \tag{13}$$

4.2.5 Communication cost

As depicted in Table 3, electronic main group distribution in OTIS-Mesh requires $\left(\sqrt{P_G} - 1\right) + (P_G - \sqrt{P_G})$ electronic links, because of the one to all broadcast communication pattern. For example, in the case of 256 processors in OTIS-Mesh, the one to all broadcast is performed among two phases, the row-wise phase and the column-wise phase. In row-wise phase, three steps are required to distribute the distance matrix to all processors within the same row of MC processor, and this requires $\sqrt{P_G} - 1$ electronic links. While in the column-wise phase, every processor received the distance matrix will send it to all processors within the same column, in parallel, that is, 12 electronic links will be utilized during this phase. This can be calculated by multiplying the number of processors in one row inside one group, by the number of processors in one row inside one group minus one. The optical distribution of data requires $P_G - 1$ optical links, which is equal to the number of groups in OTIS-Mesh minus one.

Now, in inter-group distribution of data, the number of the utilized electronic links is equal to $P_G^2 - 2P_G + 1$, that is obtained by multiplying the number of groups minus one and the number of electronic links that is utilized during the distribution inside one group. Therefore, the total communication cost of applying PRNN algorithm on OTIS-Mesh is illustrated in Eq. (14).

$$\text{Comm}_{\text{Cost}} = 2(P_G^2 - 1) \tag{14}$$

Table 5 summarizes the parallel run time complexity, speedup, efficiency, cost and communication cost of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh, where $T_{Distribution}$ is the time needed for distribution phase as a number of communication

Table 5 Parallel run time complexity, speedup, efficiency, cost and communication cost of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh

OTIS-Hypercube	$T_{\text{Distribution}}$	$\Theta\left(N^2 \times d\right)$, d is dimension of hypercube and N is number of cities
	$T_{\text{Combining}}$	$\Theta\left(N^2 - \frac{N^2}{P}\right)$, N^2 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	$E = \frac{N^3}{P^2 + N^3 + N^2 \times P \times d}$
	Cost	$\text{Cost} = P^2 + N^3 + N^2 \times d \times P$
	$\text{Comm}_{\text{Cost}}$	$\text{Comm}_{\text{Cost}} = 2\left(P_G^2 - 1\right)$, P_G is the number of processors inside one group
OTIS-Mesh	$T_{\text{Distribution}}$	$\Theta(N^2 \times \sqrt{P_G})$
	$T_{\text{Combining}}$	$\Theta\left(N^2 \times \sqrt{P_G}\right)$
	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	$E = \frac{N^3}{P^2 + N^3 + N^2 \times P \times \sqrt{P_G}}$
	Cost	$\text{Cost} = P^2 + N^3 + N^2 \times \sqrt{P_G} \times P$
	$\text{Comm}_{\text{Cost}}$	$\text{Comm}_{\text{Cost}} = 2(P_G^2 - 1)$

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, $d \ll <<2\sqrt{P_G}$; therefore, OTIS-Hypercube results are superior relative to OTIS-Mesh.

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 were 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 used a startup time equals to 55 μs [19], speed of electronic 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 was considered.

The implementation of our simulation has the following classes:

- The *distance matrix class* is responsible for reading TSPLIB instances from the .tsp files and converts them to a distance matrix that stored the distances from each city to another.
- The *repetitive nearest neighbor class* is responsible for performing the sequential nearest neighbor algorithm on each city in the distance matrix.
- The *load balancing class* is responsible for balancing N cities among P processors, where the balanced cities will be stored in the allocated cities array to be distributed to all processors.
- The *OTIS-Hypercube class* contains objects of the adopted network size ranges of OTIS-Hypercube, where the simulation starts by specifying the optoelectronic architecture followed by choosing the network size to be one of the classes A, B, C or D.
- The *node class* represents each processor and its data, such as its corresponding set of cities, the time required by each processor to find the nearest neighbor, the route matrix to store the generated routes and the cost matrix to store the cost of each route.
- The *OTIS-Mesh class* contains objects of the adopted network size ranges 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 electronic links will be determined.

Table 6 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 6. 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 6 represents a size range, where the values were chosen in a way that minimize the gap of sizes in each range between optoelectronic architectures.

To conduct the simulation runs, the traveling 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 7 shows the chosen TSP data instances from both symmetric and VLSI data sets for these simulation runs. The

Table 6 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

Table 7 TSPLIB data instances [17]

Data set	Data instance	Number of cities	Optimal solution
TSP Sym	rl1304	1304	252948
VLSI	dca1389	1389	5085
TSP Sym	u1817	1817	57201
TSP Sym	djb2036	2036	6197
VLSI	xqc2175	2175	6830
VLSI	pcb3038	3038	137694
VLSI	xqe3891	3891	11995
TSP Sym	bgb4355	4355	12723
VLSI	rl5934	5934	556045
VLSI	xsc6880	6880	21535
VLSI	bnd7168	7168	21834
VLSI	ida8197	8197	22338
VLSI	dga9698	9698	27724
TSP Sym	xmc10150	10150	28387
VLSI	rl11849	11849	923473
TSP Sym	xvb13584	13584	37084
VLSI	d15112	15112	1573084
VLSI	frh19289	19289	55798

size of these data instances is varying from 1304 cities to 19,289 cities to test the PRNN algorithm on small, medium and large number of cities.

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 7. Figures from 8–16 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

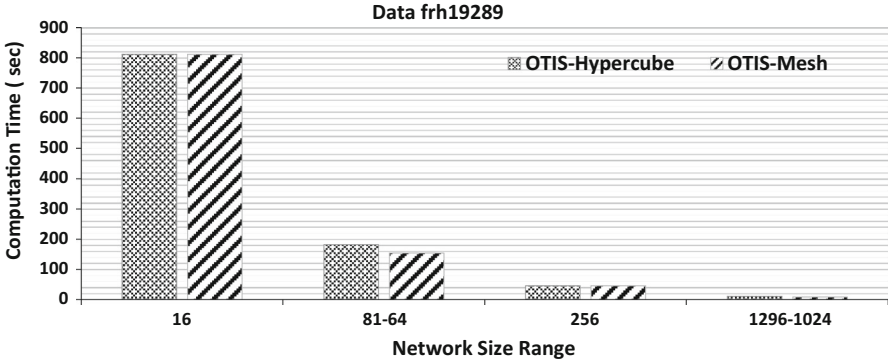

Fig. 8 Computation time of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh for frh19289 instance

for each city among *N/P* cities, as it should be obvious in Fig. 8. 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.

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 s, while PRNN algorithm on OTIS-Hypercube required 11.1 s. Consequently, PRNN algorithm on OTIS-Hypercube gave the worst computational time compared with PRNN algorithm on OTIS-Mesh, since it has less or an equal number of processors in each class.

Table 8 demonstrates the distribution time, combining time and total communication time for the PRNN algorithm on the two optoelectronic architectures with the 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 9. Note that, PRNN algorithm on OTIS-Hypercube was superior in class B, since 7 communication steps were 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

Table 8 Distribution, combining and total communication time for PRNN algorithm on OTIS-Hypercube and OTIS-Mesh with different size ranges for frh19289 TSP instance

Ranges	OTIS-Mesh	OTIS-Hypercube
Distribution time (s)		
16	46.5	46.5
64–81	91.9	69.2
256	137.4	91.9
1024–1296	228.2	114.7
Combining time (s)		
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 (s)		
16	59.6	59.6
64–81	110.5	81.8
256	161.4	103.2
1024–1296	263.2	126.0

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 a 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.

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

Table 9 Electronic 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)	
	Electronic move	OTIS move	Electronic move	OTIS move	Electronic move	OTIS move	Electronic move	OTIS move
OTIS-Hypercube	4	1	6	1	8	1	10	1
OTIS-Mesh	4	1	8	1	12	1	20	1

the main group will start 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 $N \times N/P \times (P-1)$ size. Clearly, there are three factors influencing the combining phase, which are the number of processors, communicated message size $N \times N/P$ and the number of communication steps, which depends on the diameter of the optoelectronic architecture.

The combining time in Table 8 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, 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. The 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 number of communication steps from one dimension to another in this optoelectronic architecture. So, with the augmentation in the number of processors, this difference will influence the behavior of the combining time. In general, the diameter becomes worse as OTIS-Mesh size increases and this leaves a substantial impact on the number of communication steps in this optoelectronic architecture.

Figure 9 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.

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. Figure 10 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 the 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.

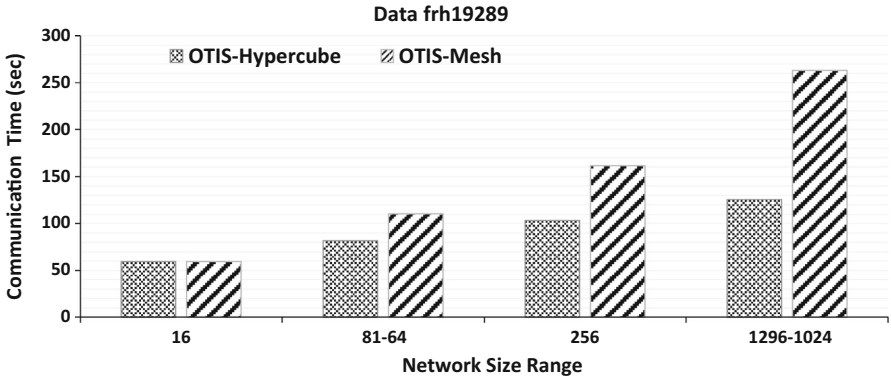

Fig. 9 Communication time of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh for frh19289 TSP instance

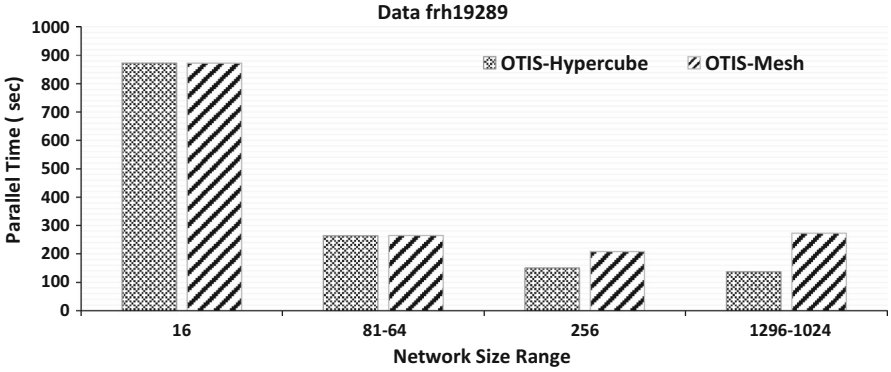

Fig. 10 Parallel time of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh for frh19289 TSP instance

Speedup results were illustrated in Fig. 11. Generally, this figure shows the growing of speedup as the number of processors increases, this growth can almost approach the number of processors. 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.

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 a 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.

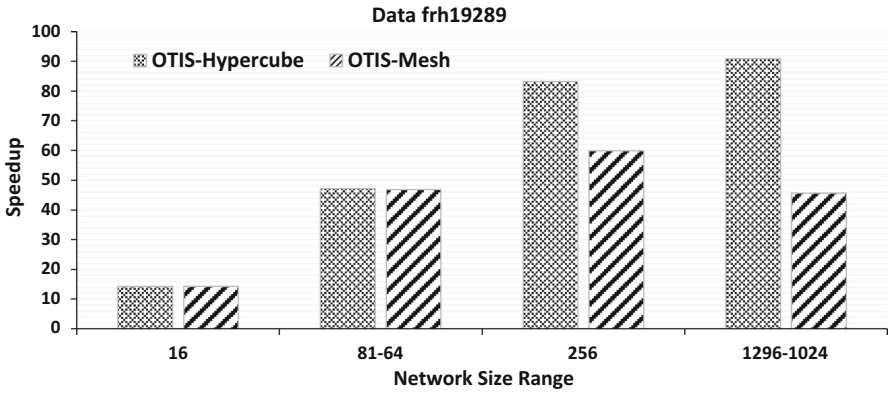

Fig. 11 Speedup of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh for frh19289 TSP instance

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.

Figure 12 depicts the efficiency of 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 the efficiency for classes B, C and D.

Figure 13 demonstrates the speedup of eighteen different TSP instances tested on OTIS-Hypercube and OTIS-Mesh in class D. They are rl1304, dca1389, u1817, djb2036, xqc2175, pcb3038, xqe3891, bgb4355, rl5934, xsc6880, bnd7168, ida8197, dga9698, xmc10150, rl11849, xvb13584, 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 7. Figure 13 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

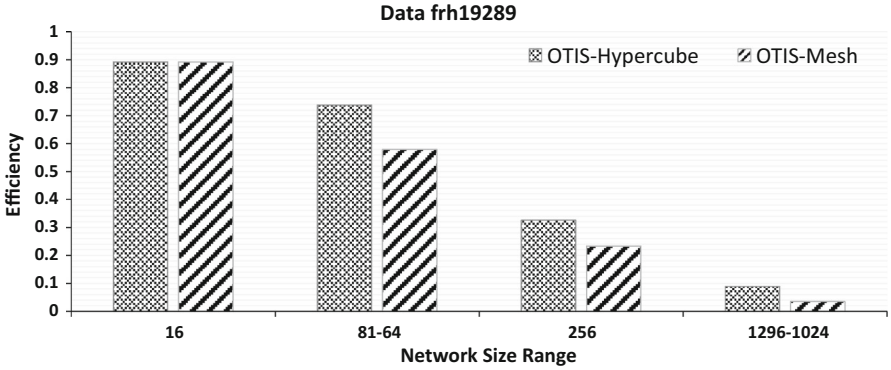

Fig. 12 Efficiency of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh for frh19289 TSP instance

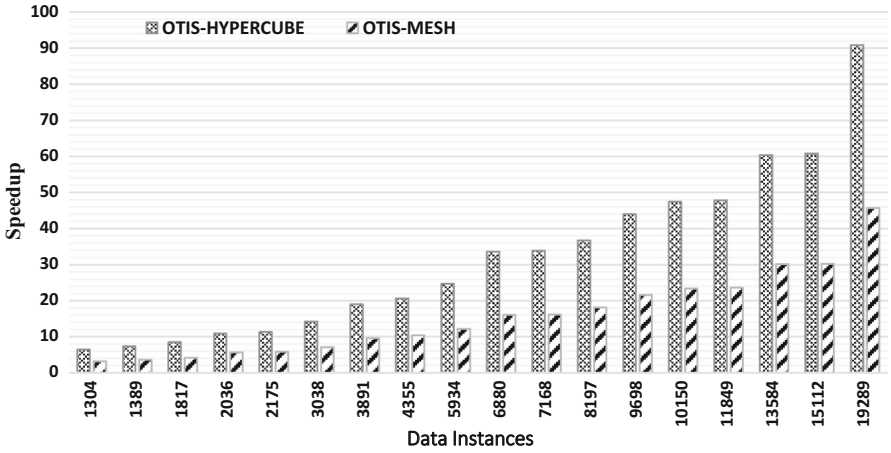

Fig. 13 Speedup of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh for class D of various TSP instances

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.

Figure 14 shows the results of applying a reduce operation rather than a gather operation on OTIS-Hypercube. As it is obvious from the figure, an improvement on the speedup was obtained using a reduce operation, since each processor only needs to forward the minimum cost route instead of gathering several routes received from other processors. This improvement is about 9% in case of 1024 processors. However, there are two reasons for adopting a gather operation rather than a reduce operation. The first reason is that if the main coordinator processor wants to improve the suboptimal solutions that were provided by the nearest neighbor algorithm, then it needs to have the generated routes, and since it is not necessary that the minimum route generated by the nearest neighbor algorithm will give the best improvement; that

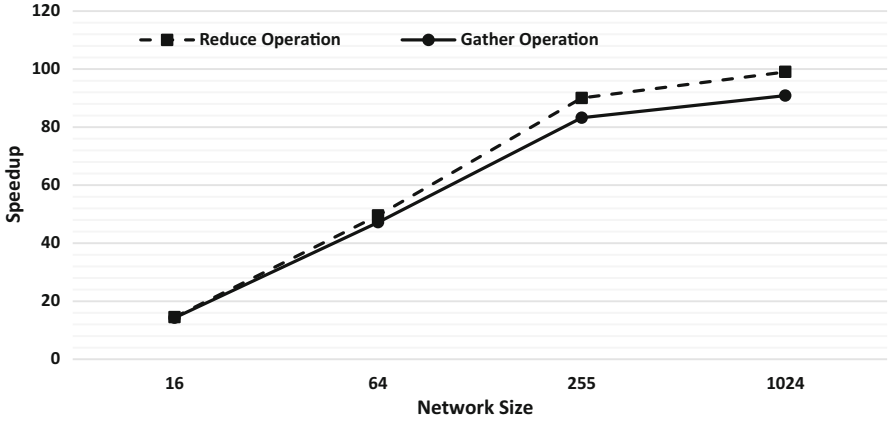

Fig. 14 Speedup of PRNN algorithm on OTIS-Hypercube for reduce and gather operations

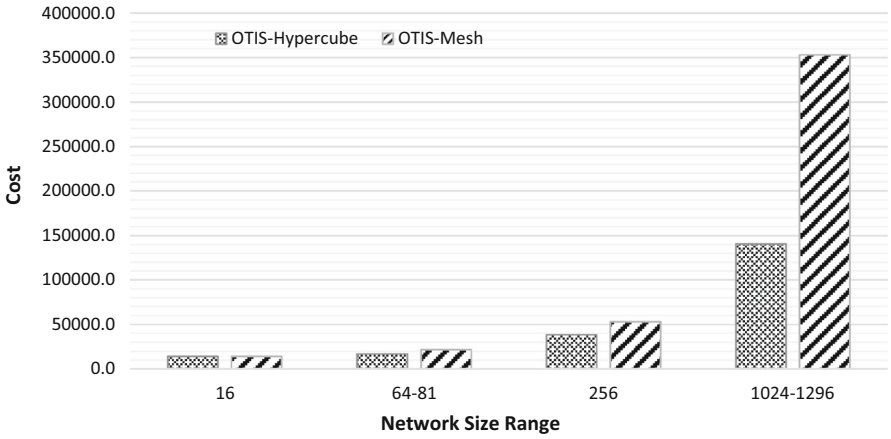

Fig. 15 Cost of PRNN algorithm on OTIS-Hypercube and OTIS-Mesh for frh19289 TSP instance

is, if it is improved by any TSP improvement algorithm, then the MC processor needs to have all the generated routes by the nearest neighbor algorithm through a gather operation. The second reason is that, if a reduce operation replaces a gather operation, then the behavior of the combining phase will be similar to the distribution phase in which it will be difficult to explain the different factors that affect the performance of combining phase. Thus, the optoelectronic architecture's performance will be assessed based only on the diameter of the basic network. For example, the performance of OTIS-Hypercube will be assessed based only on the diameter of the Hypercube. So, a performance evaluation between the selected optoelectronic architectures will not be performed as done in the discussion of the results.

Figure 15 illustrates the cost of the PRNN algorithm on OTIS-Hypercube and OTIS-Mesh, where the cost represents the total time required by all the processors in OTIS-Hypercube to apply PRNN algorithm. As it can be seen from the figure, the

Table 10 Communication cost for PRNN algorithm on OTIS-Hypercube and OTIS-Mesh

Network size range	OTIS-Hypercube			OTIS-Mesh		
	Electronic links	Optical links	Total	Electronic links	Optical links	Total
16	24	6	30	24	6	30
64–81	112	14	126	144	16	160
256	480	30	510	480	30	510
1024–1296	1984	62	2046	2520	70	2590

cost increases as the number of processors increases in the optoelectronic architecture. OTIS-Hypercube recorded less cost among all classes of size ranges, particularly in class D with less than half the cost of OTIS-Mesh. This can be explained based on the definition of the cost, which can be represented as the parallel time multiplied by the number of processors. And as illustrated previously in Fig. 10, the parallel run time for OTIS-Hypercube was less than the parallel run time for OTIS-Mesh. Therefore, multiplying less parallel time in the case of OTIS-Hypercube with less number of processors equals to 1024 processors compared to 1296 processors in the case of OTIS-Mesh, will yield this result.

Table 10 demonstrates the behavior of the communication cost as the number of processors increases. It is obvious from the table that, as the network size increases the communication cost will increase too, and this is trivial, since the increment in the number of processors will cause increment in the cost of the optoelectronic architecture, represented by both the electronic links and optical links. This table shows clearly the superior performance regarding the communication cost of OTIS-Hypercube compared to OTIS-Mesh; this can be noticed in classes B (64–81 processors) and D (1024–1296 processors) of the selected size ranges. This can relate to the fact that OTIS-Hypercube requires less communication steps and hence less communication time. Therefore, OTIS-Hypercube utilizes less number of electronic and optical links during the communication phases compared to OTIS-Mesh which recorded higher communication time.

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 Eq. (15), where S_q, F_S, O_S denote *Solution quality*, *Found Solution* and *Optimal Solution*, respectively.

$$S_q = \frac{F_S - O_S}{O_S} \times 100\% \tag{15}$$

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 the 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 11, you can notice

Table 11 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
dca1389	5085	6080	6293.7	6631	19.6	23.8	30.4	781
u1817	57201	66187	70160.0	73922	15.7	22.7	29.2	418
djb2036	6197	7645	7899.4	8274	23.4	27.5	33.5	552
xqc2175	6830	8291	8652.2	8968	21.4	26.7	31.3	1070
pcb3038	137694	169009	173377.9	179015	22.7	25.9	30.0	2198
xqe3891	11995	14592	14998.7	15690	21.7	25.0	30.8	372
bgb4355	12723	15623	16052.1	16612	22.8	26.2	30.6	2231
rl5934	556045	657056	676459.0	702041	18.2	21.7	26.3	5417
xsc6880	21535	26243.0	26948.6	27963	21.9	25.1	29.8	1056
bnd7168	21834	26574	27282.1	28199	21.7	25.0	29.2	1056
ida8197	22338	27513	28278.9	28964	23.2	26.6	29.7	5473
dga9698	27724	33564	35151.0	34331	21.1	26.8	23.8	1279
xmc10150	28387	34147	35003.7	35989	20.3	23.3	26.8	6438
rl11849	923473	1100013	1126208.9	1147853	19.1	22.0	24.3	5849
xvb13584	37084	45835	46669.2	47740	23.6	25.8	28.7	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

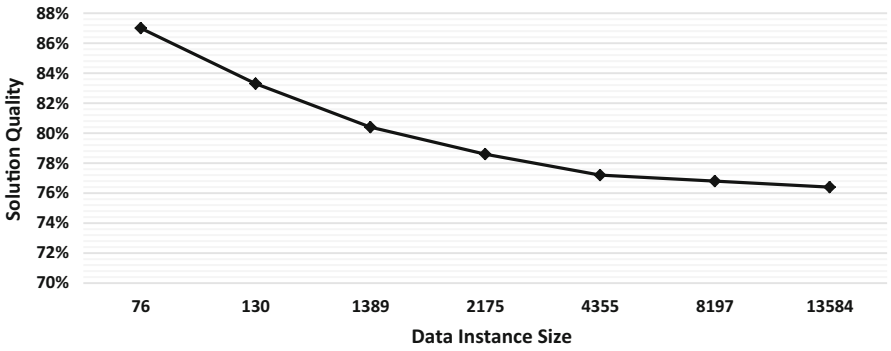

Fig. 16 Solution quality degradation as data size increases

that the algorithm gave better solutions, for example, the percentage deviation of the best solutions gave an average of 21.2% approximate solution within the optimal solution. The percentage deviation of the average solutions gave an average of 25% approximate solution within the optimal solution. Applying any random initial city such as the case of the sequential nearest neighbor gave in the worst-case an average of 29.1% approximate solutions within the optimal solution. With respect to the solution quality, the solution quality of the majority of heuristic algorithms becomes 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. Moreover, Fig. 16 shows this fact, by illustrating how the solution quality decreases as data size increases. Note that in this figure, the best solution quality is represented as how far it is from 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, but depends upon the data size; therefore, the obtained solutions were similar on both optoelectronic architectures for all size ranges.

6 Conclusions and future work

In summary, this paper introduced and evaluated PRNN algorithm for solving symmetric TSP on two selected OTIS optoelectronic architectures namely: OTIS-Hypercube and OTIS-Mesh. We discussed in detail 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. A comparative evaluation is presented as a 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. We evaluated the performance of the PRNN

algorithm over OTIS-Hypercube and OTIS-Mesh in terms of the number of communication steps, parallel run time, speedup, efficiency, cost and communication cost. In general, the number of processors in the optoelectronic architecture has a great effect on the computation time of the PRNN algorithm. For example, PRNN algorithm on OTIS-Mesh recorded better computation time than on OTIS-Hypercube, since OTIS-Mesh has equal or higher number of processors in each class of the network size ranges. Regarding the communication time, particularly the distribution phase, the diameter of the optoelectronic architecture played the exclusive role, where in the case of OTIS-Mesh, 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, compared to OTIS-Hypercube. On the other hand, the combining phase was affected by different factors, such as the diameter of the factor network that exists in each group, the number of communication steps, the communicated message size and the number of processors in each optoelectronic architecture.

The analytical evaluation results show that the parallel run time of PRNN algorithm over OTIS-Hypercube equals to $\Theta\left(P+\frac{N^3}{P}+N^2\times d\right)$, and over OTIS-Mesh equals to $\Theta(P+\frac{N^3}{P}+N^2\times\sqrt{P_G})$, which shows that the parallel run time of PRNN algorithm over OTIS-Hypercube is better than over OTIS-Mesh.

The simulation results achieved high speedup among the two optoelectronic architectures of interest. It was clear from the simulation results that PRNN algorithm on OTIS-Hypercube gained the best results in terms of communication time, parallel time, speedup, efficiency, cost and communication cost, in comparison with PRNN on OTIS-Mesh. For instance, the speedup is recorded 90.9 by OTIS-Hypercube in class D, while by OTIS-Mesh the speedup is recorded 45.7 in the same class. The results were justified based on the factors that influenced by both the computation and communication time.

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 a 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 gaining performance between them.

Acknowledgements The authors would like to express their deep gratitude to the anonymous referees for their valuable comments and helpful suggestions, which improved the paper.

References

1. Marsden G, Marchand P, Harvey P, Esener S (1993) Optical transpose interconnection system architectures. Opt Lett 18(13):1083–1085

2. Rajasekaran S, Reif J (2008) Handbook of parallel computing models, algorithms and applications. CRC Press, Boca Raton
3. Lucas KT, Jana PK (2010) Parallel algorithms for finding polynomial roots on OTIS-torus. J Supercomput 54(2):139–153
4. Jana P, Mallick D (2010) OTIS-MOT: an efficient interconnection network for parallel processing. J Supercomput 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 Distrib Syst 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 Comput 13(1):87–110
9. Mahafzah B, Jaradat B (2008) The load balancing problem in OTIS-hypercube interconnection networks. J Supercomput 46(3):276–297
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. J Supercomput 72(10):3960–3992
11. Matai R, Singh S, Mittal ML (2010) Traveling salesman problem: an overview of applications, formulations, and solution approaches. In: Davendra D (ed) Traveling salesman problem, theory and applications. InTech, pp 1–24. ISBN: 978-953-307-426-9
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. J Supercomput 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, New York, pp 1498–1506
15. Reinelt G (1994) The traveling salesman: computational solutions for TSP applications. Lect Notes Comput Sci 840:73–97
16. Zane F, Marchand P, Paturi R, Esener S (2000) Scalable network architectures using the optical transpose interconnection system (OTIS). J Parallel Distrib Comput 60(5):521–538
17. Reinelt G (1991) TSPLIB: a traveling salesman problem library. ORSA J Comput 3(4):376–384
18. Grama A, Gupta A, Karyp G, Kumar G (2003) Introduction to parallel computing. Addison Wesley, Boston
19. Hennessy JL, Patterson DA (2011) Computer architecture: a quantitative approach. Morgan Kaufmann, Burlington
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. Oper Res 37(6):865–892