An algorithm for embedding Turán graphs into incomplete hypercubes with minimum wirelength

A. Arul Shantrinal\textsuperscript{a}, Sandi Klavžar\textsuperscript{b,c,d}, R. Sundara Rajan\textsuperscript{a}, T.M. Rajalaxmi\textsuperscript{e}

\textsuperscript{a}Department of Mathematics, Hindustan Institute of Technology and Science, Chennai - 603 103, India

\textsuperscript{b}Faculty of Mathematics and Physics, University of Ljubljana, Slovenia

\textsuperscript{e}Department of Mathematics, SSN College of Engineering, Chennai 603 103, India

Abstract

The wirelength is one of the key parameters of the quality of embedding graphs into host graphs. To our knowledge, no results for computing the wirelength of embedding irregular graphs into irregular graphs are known in the literature. We develop an algorithm that determines the wirelength of embedding of the Turán graph $T(\ell, 2^p)$, where $2^{n-1} < \ell < 2^n$ and $1 \leq p \leq \lceil \log_2 \ell \rceil$, into the incomplete hypercube $I_\ell^p$. The latter graphs form an important generalization of hypercubes because they eliminate the restriction on the number of nodes in a system.

Keywords: Embedding, wirelength, Turán graph, incomplete hypercube

1 Introduction

To establish parallel systems, various interconnected schemes have been proposed. It is much desirable that such a scheme admits construction in any size and offers incremental flexibility to maximum level. One of the most popular interconnected scheme is the binary hypercube and many machines based on this topology are available. The hypercube topology, however, interconnects precisely $2^n$ nodes for some positive integer $n$, thus severely limiting permissible system sizes.

In order to eliminate the restriction on the size of the network, that is, to be able to construct machines of arbitrary sizes, incomplete hypercubes were proposed [10]. The intrinsic structure of the incomplete hypercubes involves a copy of large hypercube as well as hierarchically smaller hypercubes, and is thus eligible to simultaneously execute multiple jobs of different sizes [19, 21]. Katseff developed algorithms for routing and broadcasting messages in incomplete hypercubes [10], while some structural properties of incomplete hypercubes have been studied in [21].

A connected graph can be used to frame the topological structure of an interconnection network, but there are various equally incompatible condition in designing it. Designing an optimum network is not possible. Analyzing existing networks to be embedded in this network is a central issue in designing and evaluating an interconnection network and vice-versa [16]. Mapping a logical graph into a host graph via graph embedding is a major technique. This technique in particular allows numerous applications, let us mention architecture simulations as well as processor allocations.

There are certain cost criteria to measure the quality of an embedding. Among the most considerable criteria are the congestion and the wirelength [16]. The former one is defined as the
cardinality of a largest set of edges from the guest graph that are mapped on a single edge from the host graph. The key point here is that when we are faced with a large congestion, many problems are possible. These include, among other issues, circuit switching, long communication delay, as well as the existence of uncontrolled noise. In data networking, typical effects include packet loss or blocking of new connections. Therefore, a minimum congestion is utmost desirable in network embedding [17]. The other measure, the wirelength, is the sum of the congestions. Sources for its interest include VLSI design, data structures, and more [17]. Graph embeddings have been thoroughly investigated for a variety of networks, in particular incomplete hypercubes as a host network [2, 5, 8, 9, 20]. For additional aspects of network embeddings see [7, 12, 23].

Fully connected networks correspond to complete graphs. The latter graphs naturally generalize to complete p-partite graphs, in which the node/vertex set can be partitioned into p independent sets and there are all possible edges between vertices from different sets, cf. [17, 18]. If the parts are of cardinality \( n_i, i \in [p] \), then the complete multipartite graph is denoted by \( K_{n_1,\ldots,n_p} \). Among these graphs, Turán graphs play a prominent role. If \( n \) and \( p \) are positive integers, then the Turán graph \( T(n, p) \) is the unique \( p \)-multipartite graph of order \( n \) such that the cardinalities of its parts are as equal as possible (that is, every two cardinalities are either equal or differ by exactly one). The original source for Turán graphs lies in extremal graph theory, but they are important also elsewhere, cf. [15].

To our knowledge, no results for computing the exact wirelength of embedding irregular graphs into irregular graphs are known in the literature. In this paper, we overcome this by taking the Turán graph \( T(\ell, 2^p) \), where \( 2^n - 1 < \ell < 2^n \) and \( 1 \leq p \leq \lfloor \log_2 \ell \rfloor \), as a guest graph, and the incomplete hypercube \( I_n^\ell \) as a host graph. In the next section we formally define concepts needed and recall two key lemmas for our algorithm, the so-called Modified Congestion Lemma and the Partition Lemma. In Section 3 the algorithm is presented and its correctness proved. Two auxiliary algorithms are stated prior it. In the first, vertices of \( T(\ell, 2^p) \) are gives a special labeling, and in the second, the recursive structure of sub-hypercubes of \( I_n^\ell \) is determined.

2 Preliminaries

If \( n \geq 1 \), then the set \{1, \ldots, n\} with be denoted by \([n]\). If \( X \subseteq V(G) \), then the subgraph of \( G \) induced by \( X \) will be denoted by \( G[X] \).

Let \( G = (V(G), E(G)) \) and \( H = (V(G), E(G)) \) be graphs. An embedding \( \phi = (f, P_f) \) of \( G \) into \( H \) consists of

1. a one-to-one map \( f : V(G) \rightarrow V(H) \), and
2. a map \( P_f \) that assigns to every \( uv \in E(G) \) a \( u,v \)-path \( P_f(uv) \) in \( H \).

For brevity, we will denote in the rest of the paper the pair \((f, P_f)\) simply as \( f \). If \( f \) is an embedding of \( G \) into \( H \) and \( e_H \in E(H) \), then let \( EC_f(e_H) = |\{ e_G \in E(G) : e_H \in E(P_f(e_G)) \}| \). The congestion of \( f \) is
\[
EC_f(G, H) = \max_{e_H \in E(H)} EC_f(e_H)
\]
and the congestion of embedding \( G \) into \( H \) is
\[
EC(G, H) = \min_{f:G \rightarrow H} EC_f(G, H).\]

Further, if \( f \) is an embedding of \( G \) into \( H \) and \( S \subseteq E(H) \), then we set \( EC_f(S) = \sum_{e_H \in S} EC_f(e_H) \).
The wirelength of an embedding \( f \) of \( G \) into \( H \) is
\[
WL_f(G, H) = \sum_{e_h \in E(H)} EC_f(e_h),
\]
and the wirelength of embedding \( G \) into \( H \) is
\[
WL(G, H) = \min_{f:G \to H} WL_f(G, H).
\]
If \( G \) is a graph and \( M \subseteq V(G) \), then
\[
I_G(M) = \{uv \in E \mid u, v \in M\} \quad \text{and} \quad I_G(k) = \max_{M \subseteq V(G), |M|=k} |I_G(M)|.
\]
The maximum subgraph problem (MSP) for \( k \in [n] \) is to determine \( M \subseteq V(G) \) such that \( |M|=k \) and \( |I_G(M)| = I_G(k) \). Such a set is called an optimal set \([1, 6] \). Recall that the famous Turán’s theorem asserts that among the graphs of order \( n \) with no subgraph \( K_{p+1} \), the Turán graph \( T(n, p) \) has the maximum number of edges. For our purposes the most important consequence is that if \( G \) is a complete \( p \)-multipartite graph and \( H \) is its Turán \( p \)-partite subgraph, then \( V(H) \) is an optimal set.

The next two lemmas are efficient techniques to find the exact wirelength using MSP.

**Lemma 2.1.** \([14] \) Let \( f : G \to H \) be an embedding with \( |V(G)| = |V(H)| \). Let \( S \subseteq E(H) \) be such that \( E(H) \setminus S \) has exactly two subgraphs \( H_1 \) and \( H_2 \), and set \( G_1 = G[f^{-1}(V(H_1))] \) and \( G_2 = G[f^{-1}(V(H_2))] \). Furthermore, let \( S \) satisfy the following conditions:

(i) For every \( uv \in E(G_i), i \in [2] \), the path \( P_f(uv) \) has no edges in \( S \).

(ii) For every \( uv \in E(G), u \in V(G_1), v \in V(G_2) \), the path \( P_f(uv) \) has exactly one edge in \( S \).

(iii) \( V(G_1) \) and \( V(G_2) \) are optimal sets.

Then \( EC_f(S) \) is minimum over all embeddings \( f : G \to H \) and
\[
EC_f(S) = \sum_{v \in V(G_1)} deg_G(v) - 2|E(G_1)| = \sum_{v \in V(G_2)} deg_G(v) - 2|E(G_2)|.
\]

**Lemma 2.2.** \([14, 18] \) Let \( f : G \to H \). If \( \{P_1, \ldots, P_t\} \) is a partition of \( E(H) \), where each part \( P_i \) is an edge cut that satisfies the conditions of Lemma 2.1, then
\[
WL_f(G, H) = \sum_{i=1}^{t} EC_f(P_i).
\]
Moreover, \( WL(G, H) = WL_f(G, H) \).

If \( G \) is a connected graph, then the relation \( \Theta \) is a defined on \( E(G) \) in the following way. An edge \( e = xy \) is in relation \( \Theta \) with an edge \( f = uv \) if and only if \( d(x, u) + d(y, v) = d(x, v) + d(y, u) \). It is straightforward to see that \( \Theta \) is both reflexive and symmetric. Consequently, the transitive closure \( \Theta^\ast \) of \( \Theta \) is an equivalence relation. The corresponding partition is called the \( \Theta^\ast \)-partition of \( E(G) \), see [4, 11, 22].

We still need to define the incomplete hypercubes \([3] \). If \( 2^{n-1} < \ell < 2^n \), then the \( n \)-dimensional incomplete hypercube \( I_\ell^n \) has \( \ell \) vertices and is defined recursively as follows. \( I_\ell^n \) comprises two components, \( Q_{n-1} \) and \( I_{\ell-2^{n-1}}^k \), where \( k = \lfloor \log_2(\ell - 2^{n-1}) \rfloor \). The vertices of \( Q_{n-1} \) are numbered from 0 to \( 2^{n-1} - 1 \), and the vertices in \( I_{\ell-2^{n-1}}^k \) from \( 2^{n-1} \) to \( \ell - 1 \) using lexicographic ordering \([13] \). An edge exists between a vertex \( u \in V(Q_{n-1}) \) with label \( i \) and a vertex \( v \in V(I_{\ell-2^{n-1}}^k) \) with label \( j \) if and only if \( |i - j| = 2^{n-1} \). In Fig. 1 the incomplete hypercube \( I_5^{23} \) is drawn.
3 The algorithm

In this section we describe an algorithm that computes the minimum wirelength of embedding $T(\ell, 2^p)$ into $I_n^\ell$, where $2^{n-1} < \ell < 2^n$ and $1 \leq p \leq \lceil \log_2 \ell \rceil$. For this sake we first have a closer look to incomplete hypercubes.

The complete hypercube $I_n^\ell$ contains a set of hypercubes of dimension $n - 1$ and below, where no two cubes have the same cardinality. For instance, $I_5^{23}$ (Fig. 1) comprises $Q_4$ and $I_2^3$, which, in turn, contains $Q_2$ and $I_2^3$. If the binary representation of $\ell$ is $1x_{n-2} \ldots x_1 x_0$, then $I_n^\ell$ contains, in addition to $Q_{n-1}$, also $Q_i$, for all $0 \leq i \leq n - 2$ for which we have $x_i = 1$. That is, $Q_i$ is a cube of $I_n^\ell$ iff the bit $x_i$ in the binary representation of $\ell$ equals 1. Consequently, the set of constituent cubes in an incomplete hypercube is unique [3]. In the rest we use the label $i + 1$ instead of $i$, so that the set of labels of the vertices in $I_n^\ell$ is $\{1, \ldots, 2^{n-1}, 2^{n-1} + 1, \ldots, \ell\}$.

To make the algorithm more transparent, we first consider in detail the following specific case.

3.1 Computing the wirelength of embedding $T(23, 4)$ into $I_5^{23}$

$\Theta^*$ partitions $E(I_5^{23})$ into $\Theta^*$-classes $E_i$, $i \in [5]$, see the right-hand side of Fig. 2. Note that each $E_i$ satisfies the first two conditions of Lemma 2.1. In addition, we label the vertices of $I_5^{23}$ using the lexicographic labeling, see the right-hand side of Fig. 2 again.

Note that $T(23, 4) = K_{5,6,6,6}$ and let $V_1 = \{1, 8, 9, 16, 17\}$, $V_2 = \{2, 7, 10, 15, 18, 23\}$, $V_3 = \{3, 6, 11, 14, 19, 22\}$, and $V_4 = \{4, 5, 12, 13, 20, 21\}$ be the maximal independent sets of $T(23, 4)$, see the left-hand side of Fig. 2. Define now the embedding $f$ of $T(23, 4)$ into $I_5^{23}$ with $f(k) = k$, $k \in [23]$, where for $kk' \in E(T(23, 4))$, the path $P_f(kk')$ is an arbitrary, fixed shortest $f(k), f(k')$-path in $I_5^{23}$.

$I_5^{23}\backslash E_1$ consists of components $H_1$ and $H_2$, where $V(H_1) = \{1, 2, \ldots, 8, 17, \ldots, 23\}$ and $V(H_2) = \{9, 10, \ldots, 16\}$. Let $G_1 = T(23, 4)[f^{-1}(V(H_1))$ and $G_2 = T(23, 4)[f^{-1}(V(H_2)]$. The graphs $G_1$ and $G_2$ are isomorphic to $K_{3,4,4,4}$ and to $K_{2,2,2,2}$, respectively. Since these are Turán graphs, Turán’s theorem implies that $V(G_1)$ and $V(G_2)$ are optimal sets. Thus, $E_1$ (having in mind that $H_1$ and $H_2$ induce convex subgraphs) satisfies conditions (i)-(iii) of Lemma 2.1. Consequently, $EC_f(E_1)$ is
minimun and

\[ EC_f(E_i) = \sum_{v \in V(G_1)} \deg_G(v) - 2|E(G_1)| = 90. \]

Using parallel arguments as above, we proceed with \( I_5^{23} \backslash E_i, i \in \{2, 3, 4, 5\} \). First, \( I_5^{23} \backslash E_2 \) consists of components \( H_1 \) and \( H_2 \), where \( V(H_1) = [16] \) and \( V(H_2) = [23] \backslash [16] \). Then \( T(23, 4)[f^{-1}(V(H_1))] = T(16, 4) \) and \( T(23, 4)[f^{-1}(V(H_2))] = T(7, 4) \), hence \( EC_f(E_2) \) is minimum and

\[ EC_f(E_2) = \sum_{v \in V(G_1)} \deg_G(v) - 2|E(G_1)| = 84. \]

Next, \( I_5^{23} \backslash E_3 \) consists of components \( H_1 \) and \( H_2 \), where \( V(H_1) = \{1, 2, 3, 4, 9, 10, 11, 12, 17, 18, 19, 20\} \) and \( V(H_2) = [23] \backslash V(H_1) \). Now we have \( T(23, 4)[f^{-1}(V(H_1))] = T(12, 4) \) and \( T(23, 4)[f^{-1}(V(H_2))] = T(11, 4) \), hence \( EC_f(E_3) \) is minimum and

\[ EC_f(E_3) = \sum_{v \in V(G_1)} \deg_G(v) - 2|E(G_1)| = 99. \]

For \( i = 4 \) we have \( V(H_1) = \{1, 2, 5, 6, 9, 10, 13, 14, 17, 18, 21, 22\} \) and \( V(H_2) = [23] \backslash V(H_1) \), the corresponding Turán graphs being \( T(12, 4) \) and \( T(11, 4) \), so that \( EC_f(E_4) \) is also minimum and

\[ EC_f(E_4) = \sum_{v \in V(G_1)} \deg_G(v) - 2|E(G_1)| = 99. \]

Finally, for \( i = 5 \) we have \( V(H_1) = \{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23\} \), \( V(H_2) = [23] \backslash V(H_1) \), and the Turán graphs \( T(12, 4) \) and \( T(11, 4) \). So \( EC_f(E_5) \) is minimum and

\[ EC_f(E_5) = \sum_{v \in V(G_1)} \deg_G(v) - 2|E(G_1)| = 99. \]
By Lemma 2.2 we conclude that

\[ WL(G, H) = WL_f(G, H) = \sum_{i=1}^{5} EC_f(E_i) = 90 + 84 + 99 + 99 + 99 = 471. \]

### 3.2 The general case

In this section we give an algorithm that computes the minimum wirelength of embedding \( T(\ell, 2^p) \) into \( I_n^\ell \), where \( 2^{n-1} < \ell < 2^n \) and \( 1 \leq p \leq \lceil \log_2 \ell \rceil \).

We start with an auxiliary algorithm that accordingly labels the vertices of \( T(\ell, 2^p) \). We thus have \( \ell \) vertices partitioned into \( m = 2^p \) parts. Initially, all the vertices are unlabeled. Then label the first vertex in each partition (up to \( m \)) by increment of 1 using clockwise direction. Then label the second vertex in the \( m^{th} \) partition as \( m + 1 \). Now, switch the numbering in anticlockwise direction. Continue this process until all the \( \ell \) vertices are labeled. The formal algorithm is given below as Algorithm 1.

**Algorithm 1:**

**Input:**
- \( m \): number of partitions.
- \( \ell \): total number of vertices.

**Output:** Labeling of the vertices of \( T(\ell, 2^p) \).

**Step 1.** Initialize \( m, n, \ell, \) and \((x, y)\), where \( n = \lceil \ell/m \rceil \) and in \((x, y)\), \( x \) represent the partition and \( y \) represent the vertex position of the partition.

**Step 2.** Get the values of \( m \) and \( \ell \) as inputs.

**Step 3.** Initialize \( i = 1, d = 1 \)

**Step 4.** for \((y \leftarrow 1 \) to \( n \), increment \( y \) by 1)

**Step 4.1.** if \( d = 1 \)

**Step 4.2.** for \((x \leftarrow 1 \) to \( m \), increment \( x \) by 1)

**Step 4.2.1.** if \( i > \ell \)

**Step 4.2.2.** Print \((x, y) = 0\)

else

**Step 4.2.3.** Print \((x, y) = i\)

**Step 4.2.4.** Increment \( i \) value by 1

**Step 4.2.5.** Multiply the value of \( d \) by (-1)

else

**Step 4.3.** for \((x \leftarrow m \) to \( 0 \), decrement \( x \) by 1)

**Step 4.3.1.** if \( i > \ell \)

**Step 4.3.2.** Print \((x, y) = 0\)

else

**Step 4.3.3.** Print \((x, y) = i\)

**Step 4.3.4.** Increment \( i \) value by 1

**Step 4.3.5.** Multiply the value of \( d \) by (-1)

**Step 5.** Repeat Step 4 until \( y > n \)

**Step 6.** Print the labeling of the Turán graph \( T(\ell, 2^p) \)
Lemma 3.1. If $2^{n-1} < \ell < 2^n$ and $1 \leq p \leq \lceil \log_2 \ell \rceil$, then Algorithm 1 labels the vertices of $T(\ell, 2^p)$ with different integers from $[\ell]$.

Proof. The graph $T(\ell, 2^p)$ has $\ell$ vertices partitioned into $m = 2^p$ independent sets. Algorithm 1 proceeds as described before. More specifically, Step 4.2 labels the first vertex of the first partition as 1, and continues the same process up to $m^{th}$ partition by increment of 1. Then, in Step 4.3, the second vertex of the $m^{th}$ partition is labeled with $m + 1$. Switching the numbering in anti-clockwise direction we label vertices by increment of 1 until we reach the $1^{st}$ partition. Step 4 is repeated until the $\ell^{th}$ vertex is reached and accordingly labeled.

An implementation of Algorithm 1 in Python and two of its outputs are listed in Annexure I.

We continue with an auxiliary algorithm that identifies the recursive structure of sub-hypercubes of $I_n^\ell$, where $2^{n-1} < \ell < 2^n$ and $n = \lceil \log_2 \ell \rceil$. The algorithm in pseudo-code reads as follows.

Algorithm 2:

Input: Positive integers $n$ and $\ell$, where $2^{n-1} < \ell < 2^n$ and $n = \lceil \log_2 \ell \rceil$.

Output: The sub-hypercubes structure of $I_n^\ell$ in an array form.

Step 1. Initialize $A[n]$, $i$, $n$, $r$, $\ell$.

Step 2. Get values $\ell, n$ and find $I_n^\ell$, where $r = \ell$ and $n = \lceil \log_2 \ell \rceil$.

Step 3. Start from the root node.

Step 4. Traverse till the end node (i.e.) when $r = 0$.

Step 5. for $(i \leftarrow 1$ to $n)$ do

Step 5.1. $A[n - i] = 2^{n-i}$

Step 5.1.1. if $(A[n - i] \leq r)$

Step 5.1.2. Print the value of $A[n - i]$

Step 5.1.3. $r = r - A[n - i]$

Step 5.1.4. end if

Step 5.2. end for

Step 6. Print the array of incomplete hypercube $I_n^\ell$.

Lemma 3.2. Algorithm 2 encodes in an array the sub-hypercubes structure of $I_n^\ell$.

Proof. We find the number of vertices of each sub-hypercube of $I_n^\ell$ using Step 5.1, that is, $A[n - i] = 2^{n-i}$, $i \in [n]$. In Step 5.1.3 we update the value of $r$ using the condition $r = r - A[n - i]$. Step 5 is repeated until $i = n$ and $r = 0$.

To illustrate Algorithm 2, first see an overview of the structure of $I_n^\ell$ as illustrated in Fig. 3. Second, consider the incomplete hypercube $I_n^{23}$. In Algorithm 2 we have $n = 5$ and $\ell = r = 23$. For $i = 1$, we get $A[4] = 2^{4} = 16$, which is the order of $Q_4$. Next $r = r - A[n - i] = 23 - 16 = 7$, and so $A[3] = 2^3 = 8$. Here $A[n - i] = 8 \notin 7 = r$. So, condition fails and the value of $r$ remains the same. For $i = 3$ we get $A[2] = 4$ Here $A[n - i] = 4 \leq 7 = r$, so $r = r - A[n - i] = 7 - 4 = 3$. For $i = 4$ we get $A[1] = 2$ and for $i = 5$ we get $A[0] = 1$. Then $r = 0$ and the algorithm ends.

Now we are ready for the main algorithm. In it we first label the vertices of $T(\ell, 2^p)$ and find the structure of sub-hypercubes of $I_n^\ell$. After that we label the vertices of $I_n^\ell$. Then we find a mapping $f : V(G) \rightarrow V(H)$ such that the $\Theta^*$-partition of $I_n^\ell$ satisfies all the conditions of the Lemma 2.1. This is formalized in Algorithm A.
Algorithm A:
Input: Positive integers $\ell, n, p$, where $2^{n-1} < \ell < 2^n$, $1 \leq p \leq \lceil \log_2 \ell \rceil$, and $n = \lfloor \log_2 \ell \rfloor$.
Output: Minimum wirelength of embedding $T(\ell, 2^p)$ into $I_n^\ell$.

**Step 1.** Label the vertices of $T(\ell, 2^p)$ using Algorithm 1.

**Step 2.** Run Algorithm 2 to determine the sub-hypergraph structure of $I_n^\ell$ and label its vertices using lexicographic labeling.

**Step 3.** Replace the label $i$ by $i + 1$ in $I_n^\ell$.

**Step 4.** For all $i \in V(G)$ let $f(i) = i$ and for all $ii' \in E(T(\ell, 2^p))$, let the path $P_f(ii')$ be an arbitrary but fixed shortest $f(i), f(i')$-path in $I_n^\ell$.

**Step 5.** Find $f : V(G) \to V(H)$ such that $\Theta^*$-partition of $I_n^\ell$ satisfies the conditions of the Lemma 2.1.
Step 6. Use Lemma 2.2 to find the exact wirelength.

Theorem 3.3. Algorithm A is correct.

Proof. The relation $\Theta^*$ partitions $E(I_n^\ell)$ into classes $E_i$, $i \in [n]$. Each of these classes fulfills the first two conditions of the Lemma 2.1. $E(I_n^\ell/E_i)$, $i \in [n]$, has exactly two components $H_{i1}$ and $H_{i2}$, where the cardinalities of $V(H_{i1})$ and $V(H_{i2})$ depend on the recursive structure of $I_n^\ell$ and can be deduced from Algorithm 2. If $G_1 = T(\ell, 2^p)[f^{-1}(V(H_{i1}))]$ and $G_2 = T(\ell, 2^p)[f^{-1}(V(H_{i2}))]$, then $G_1$ and $G_2$ are Turán graphs and Turán theorem implies that $V(G_1)$ and $V(G_2)$ are optimal sets. Thus, the $\Theta^*$-equivalence class $E_i$, $i \in [n]$, satisfies the third condition of the By Lemma 2.1. Therefore, $EC_f(E_i)$ is minimum for all $i \in [n]$. By Lemma 2.2 we conclude that the wirelength is minimum.

4 Concluding remarks

Finding the wirelength of embedding Turán graphs into further architectures such as grids, arbitrary trees, Christmas trees, hypertrees, Cayley graphs, and permutation graphs, are under investigation.

Acknowledgments

We thank G. Kirithiga Nandini, Junior Research Fellow, Hindustan Institute of Technology and Science, Chennai, India; T. Revathi, Research Scholar, SSN College of Engineering, Chennai, India and S. Renish, Software Associate, Capgemini Technology Services India Ltd, Chennai for their suggestions.

This work was supported by Project No. 2/48(4)/2016/NBHM-R&D-II/11580, National Board of Higher Mathematics (NBHM), Department of Atomic Energy (DAE), Government of India, and by the Slovenian Research Agency (research core funding No. P1-0297 and projects J1-9109, N1-0095).

References


Annexure I

Python program for labeling of the guest graph:

```python
arry = [[] for i in range(100)]
i = j = k = l = m = elem = elem_val = x = -1
flag = last_flag = None
def direct():
    global i, j, k, l, m, elem, elem_val, x, flag, last_flag, arry
    for k in range(x):
        elem_val += 1
        if(elem_val > elem):
            break
        arry[k][i * m + j] = elem_val
def reverse():
    global i, j, k, l, m, elem, elem_val, x, flag, last_flag, arry
    for k in range(x-1,-1,-1):
        elem_val += 1
        if(elem_val > elem):
            break
        arry[k][i * m + j] = elem_val
def last_reverse():
    global i, j, k, l, m, elem, elem_val, x, flag, last_flag, arry
    for j in range(m-1,-1,-1):
        if(flag == 'X'):
            direct()
            flag = 'Y'
        else:
            reverse()
            flag = 'X'
def last_direct():
    global i, j, k, l, m, elem, elem_val, x, flag, last_flag, arry
    for j in range(m):
        if(flag == 'X'):
            direct()
            flag = 'Y'
        else:
            reverse()
            flag = 'X'
def main():
    global i, j, k, l, m, elem, elem_val, x, flag, last_flag, arry
    # reading values
    print('Enter the number of elements: '),
    elem = int(input())
    print('Enter the number of array: '),
    x = int(input())
    m = elem / x
    l = elem % x
    if(l != 0):
```

11
```python
m+ = 1
l = 0
l = m%2
m = m/2
if(l! = 0):
    m+ = 1
    is dynamic array initialization
for i in range(x):
    for j in range(2 * m):
        arry[i].append(0)
    is logic for populating the values as required
elem_val = 0
flag = 'X'
last_flag = 'X'
for i in range(2):
    if(last_flag == 'X'):
        last_direct()
        last_flag = 'Y'
    else:
        last_reverse()
        last_flag = 'X'
print() for l in range(x):
    for i in range(2):
        for j in range(m):
            print('{ }'.format(arry[l][i * m + j]), print() print() if_name_ == 'main' :
        main()
```
Implementation of the above Python program

Output 1:

<table>
<thead>
<tr>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>64</td>
<td>65</td>
<td>128</td>
</tr>
<tr>
<td>129</td>
<td>192</td>
<td>193</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>63</td>
<td>66</td>
<td>127</td>
</tr>
<tr>
<td>130</td>
<td>191</td>
<td>194</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>62</td>
<td>67</td>
<td>126</td>
</tr>
<tr>
<td>131</td>
<td>190</td>
<td>195</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>61</td>
<td>68</td>
<td>125</td>
</tr>
<tr>
<td>132</td>
<td>189</td>
<td>196</td>
<td>0</td>
</tr>
<tr>
<td>V</td>
<td>VI</td>
<td>VII</td>
<td>VIII</td>
</tr>
<tr>
<td>5</td>
<td>60</td>
<td>69</td>
<td>124</td>
</tr>
<tr>
<td>133</td>
<td>188</td>
<td>197</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>59</td>
<td>70</td>
<td>123</td>
</tr>
<tr>
<td>134</td>
<td>187</td>
<td>198</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>58</td>
<td>71</td>
<td>122</td>
</tr>
<tr>
<td>135</td>
<td>186</td>
<td>199</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>57</td>
<td>72</td>
<td>121</td>
</tr>
<tr>
<td>136</td>
<td>185</td>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>IX</td>
<td>X</td>
<td>XI</td>
<td>XII</td>
</tr>
<tr>
<td>9</td>
<td>56</td>
<td>73</td>
<td>120</td>
</tr>
<tr>
<td>137</td>
<td>184</td>
<td>201</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>55</td>
<td>74</td>
<td>119</td>
</tr>
<tr>
<td>138</td>
<td>183</td>
<td>202</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>54</td>
<td>75</td>
<td>118</td>
</tr>
<tr>
<td>139</td>
<td>182</td>
<td>203</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>53</td>
<td>76</td>
<td>117</td>
</tr>
<tr>
<td>140</td>
<td>181</td>
<td>204</td>
<td>0</td>
</tr>
<tr>
<td>XIII</td>
<td>XIV</td>
<td>XV</td>
<td>XVI</td>
</tr>
<tr>
<td>13</td>
<td>52</td>
<td>77</td>
<td>116</td>
</tr>
<tr>
<td>141</td>
<td>180</td>
<td>205</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>51</td>
<td>78</td>
<td>115</td>
</tr>
<tr>
<td>142</td>
<td>179</td>
<td>206</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>79</td>
<td>114</td>
</tr>
<tr>
<td>143</td>
<td>178</td>
<td>207</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>49</td>
<td>80</td>
<td>113</td>
</tr>
<tr>
<td>144</td>
<td>177</td>
<td>208</td>
<td>0</td>
</tr>
<tr>
<td>XVII</td>
<td>XVIII</td>
<td>XIX</td>
<td>XX</td>
</tr>
<tr>
<td>17</td>
<td>48</td>
<td>81</td>
<td>112</td>
</tr>
<tr>
<td>145</td>
<td>176</td>
<td>209</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>47</td>
<td>82</td>
<td>111</td>
</tr>
<tr>
<td>146</td>
<td>175</td>
<td>210</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>46</td>
<td>83</td>
<td>110</td>
</tr>
<tr>
<td>147</td>
<td>174</td>
<td>211</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>45</td>
<td>84</td>
<td>109</td>
</tr>
<tr>
<td>148</td>
<td>173</td>
<td>212</td>
<td>0</td>
</tr>
<tr>
<td>XXI</td>
<td>XXII</td>
<td>XXIII</td>
<td>XXIV</td>
</tr>
<tr>
<td>21</td>
<td>44</td>
<td>85</td>
<td>108</td>
</tr>
<tr>
<td>149</td>
<td>172</td>
<td>213</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>43</td>
<td>86</td>
<td>107</td>
</tr>
<tr>
<td>150</td>
<td>171</td>
<td>214</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>42</td>
<td>87</td>
<td>106</td>
</tr>
<tr>
<td>151</td>
<td>170</td>
<td>215</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>41</td>
<td>88</td>
<td>105</td>
</tr>
<tr>
<td>152</td>
<td>169</td>
<td>216</td>
<td>0</td>
</tr>
<tr>
<td>XXV</td>
<td>XXVI</td>
<td>XXVII</td>
<td>XXVIII</td>
</tr>
<tr>
<td>25</td>
<td>40</td>
<td>89</td>
<td>104</td>
</tr>
<tr>
<td>153</td>
<td>168</td>
<td>217</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>39</td>
<td>90</td>
<td>103</td>
</tr>
<tr>
<td>154</td>
<td>167</td>
<td>218</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>38</td>
<td>91</td>
<td>102</td>
</tr>
<tr>
<td>155</td>
<td>166</td>
<td>219</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>37</td>
<td>92</td>
<td>101</td>
</tr>
<tr>
<td>156</td>
<td>165</td>
<td>220</td>
<td>0</td>
</tr>
<tr>
<td>XXIX</td>
<td>XXXX</td>
<td>XXXI</td>
<td>XXXII</td>
</tr>
<tr>
<td>29</td>
<td>36</td>
<td>93</td>
<td>100</td>
</tr>
<tr>
<td>157</td>
<td>164</td>
<td>221</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>35</td>
<td>94</td>
<td>99</td>
</tr>
<tr>
<td>158</td>
<td>163</td>
<td>222</td>
<td>227</td>
</tr>
<tr>
<td>31</td>
<td>34</td>
<td>95</td>
<td>98</td>
</tr>
<tr>
<td>159</td>
<td>162</td>
<td>223</td>
<td>226</td>
</tr>
<tr>
<td>32</td>
<td>33</td>
<td>96</td>
<td>97</td>
</tr>
<tr>
<td>160</td>
<td>161</td>
<td>224</td>
<td>225</td>
</tr>
</tbody>
</table>

Figure 4: Complete 32-partite graph $K_{7,...,7,8,8,8}$ with 227 nodes

Note: The label ‘0’ is considered as an empty element.
Figure 5: Complete 8-partite graph $K_{29,29,29,\ldots,29}$ with 227 nodes

Note: The label '0' is considered as an empty element.