Improving In-Network Computing in IoT Through Degeneracy

Merim Dzaferagic *, Neal McBride *, Ryan Thomas †, Irene Macaluso *, and Nicola Marchetti *
(dzaferam@tcd.ie, mcbridne@tcd.ie, ryan.thomas@usafa.edu, macalusi@tcd.ie, nicola.marchetti@tcd.ie)
*CONNECT - Trinity College Dublin, Ireland
†AFOSR, Air Force Office of Scientific Research, USA

***Abstract*—We present a novel way of considering in-network computing (INC), using ideas from statistical mechanics. We model the execution of a distributed computation with graphs called functional topologies, which allows us to provide a formal definition for degeneracy and redundancy in the context of INC. Degeneracy for INC is defined as the structural multiplicity of possible options available within the network to perform the same function with a given macroscopic property (e.g. delay). Two degenerate structures can partially overlap. Redundancy, on the other hand, does not allow overlapping between the functional graphs. We present an efficient algorithm to determine all these multiple options and compute both degeneracy and redundancy. Our results show that by exploiting the set of possible degenerate alternatives, we can significantly improve the successful computation rate of a symmetric function, while still being able to satisfy requirements such as delay or energy consumption.**

***Index Terms*—In-network computing, distributed computing, degeneracy, redundancy, IoT**

I. Introduction

The purpose of traditional data networks such as the Internet is to enable end-to-end information transfer. Information streams in such networks are carried across point-to-point links in which intermediate nodes forward data packets without meaningfully modifying their payloads. Internet of Things (IoT) networks, however, perform actions or make decisions powered by access to raw or processed data gathered by spatially distributed "things". Distributed In-Network Computation (INC) approaches, i.e. the processing of raw data within the IoT network, are becoming increasingly popular in that they can potentially achieve higher energy efficiency, lower computational delay, and higher robustness compared to the traditional approach, which involves transmitting raw data to the sink and then performing the computation.

As IoT networks become increasingly larger, their intercommunication becomes complex enough to resemble a thermodynamic system of many interacting objects. This situation leads naturally to a physics-inspired study of their behavior using statistical mechanics and a concept known as degeneracy. Degeneracy arises from structurally different configurations of a system (microstates) having functionally similar/identical macroscopic properties (macrostates). A concise definition of degeneracy comes from a modern review of the use of the term: "degeneracy describes the ability of different structures to be conditionally interchangeable in their contribution to system functions" [1]. In the context of INC, we view these different structures as functional topologies [2]: subsets of the network which enable the computation of a distributed function. In this paper, we consider degeneracy as the multiplicity of available functional topologies that enable distributed INC with a given macroscopic observable property, i.e., delay.

A. Related work

The intended application of an IoT network usually defines the type of functions being considered. This in turn naturally defines the observables (macrostates) of interest to analyze these functions. For example, in alarm networks, the quantity of interest is the threshold value (i.e., max or min) of the sensor readings, whereas in environmental monitoring, a relevant statistic would be the mean or the trend measurements (i.e., increase, decrease or oscillation over time). The distributed computing literature [3]–[7] usually assumes an error free scenario, i.e. no node failures, and mainly focuses on the optimization of the computational rate and the communication cost (e.g. energy, delay). On the other hand, we are interested in a more realistic network setup, in which node failures do exist, and we study the impact of degenerate computation and communication paths on the success rate of the computation.

Authors like [3]–[9] address communication aspects of distributed in-network computation. They focus on the optimization of the communication/computation parameters, like computational complexity, energy efficiency, computational throughput and computational delay. In contrast, we focus on the degeneracy of the network by finding the multiple feasible computational graphs that satisfy a given requirement (e.g. delay), rather than a single optimal computational graph. This way, it is possible to fully harness the computational capability of the network and significantly increase the robustness of the computation, while still being able to satisfy requirements, such as delay or energy consumption.

To date, the INC literature features analysis of functions such as distributed Fast Fourier Transform, Singular-Value Decomposition [10], MapReduce and Dryad [5], collaborative compressed sensing [11], distributed neural networks [8], and Kalman filtering [12]. The majority of IoT applications, however, rely on relatively simple aggregate functions like max, min, count and average [3], [9], [13]–[17]. The distributed computation of these functions is usually modeled with tree structures, e.g. Steiner trees [3], [4]. Closely related to the approach proposed in our work is the standard Steiner Tree Packing problem[1] [18] and its well know relaxation, i.e. the Fractional Steiner Tree Problem. Instead of using the standard Steiner Tree Packing, we rather extend the search

[1]The Steiner Tree Packing problem is to find the maximum number of edge-disjoint subgraphs of a given graph that connect a given set of required points.

from minimum weight Steiner trees to all existing Steiner trees in which the sum of the weights on the links is lower than a predefined value. Due to the need to discover all trees that satisfy a set of requirements rather than searching for the minimum weight trees, the search space is expanded when compared to the standard Steiner Tree Packing problem.

Closely related to the problem discussed in this paper is the Virtual Network Embedding problem. Network virtualization, as one of the most promising technologies for future networks, enables the dynamic instantiation of virtual/functional nodes throughout the network [19]. As emphasized by [20], the Virtual Network Embedding problem, the embedding of virtual networks in a physical topology, is the main resource allocation challenge in network virtualization. This involves using embedding algorithms that optimize the resource discovery and allocation to satisfy different embedding objectives (e.g. provide QoS-compliant embeddings, maximize the profit of the infrastructure providers, provide resilient virtual network embeddings). The Virtual Network Embedding problem is, among others, being applied to Service Chain Provisioning for Network Function Virtualization in communication networks; and optimization of the virtual node placement for Cloud Service Providers, Application Service Providers and Internet Service Providers. Unlike the usual approach taken in the literature [21]–[24], which focuses on the matching of the physical resources with the demands of the virtual networks, our approach builds on top of those studies and focuses on the analysis of the structure that connects the physical resources. By studying the physical connectivity between the network resources, while considering all communication constraints that affect the interoperability between them, we are able to quantify the size of the feasible space for the embedding problem.

B. Motivation and Contributions

As discussed in the previous section, the literature on INC and Virtual Network Embedding focus on the embedding of functions on top of the physical topology. In this paper we focus on the study of degeneracy, which defines the cardinality of the feasibility space for the embedding problem. Therefore, degeneracy can be used to evaluate the potential of a physical network to accommodate embeddings. In addition to evaluating a deployment, degeneracy could also be used to assess potential changes to the network topology. For example, in case of physical node rewiring, only the rewiring that will result in higher degeneracy should be allowed.

The main contributions of this work are:

- We propose an efficient algorithm that generates all functional topologies satisfying a given delay requirement, for any physical topology;
- We analyze the time complexity of the proposed algorithm, and its applicability to realistic network topologies;
- We provide a formal definition and calculation of degeneracy and the related concept of redundancy in terms of distributed computing over IoT networks;
- We provide a comparison of degenerate INC and a traditional multi-hop scheme that selects a single graph for the computation showing that it is possible to significantly increase the probability of successful computation by exploiting degeneracy.

II. Framework

We define a physical network $G = (V, E)$, as a simple graph that contains no self-edges or repeated edges, where the set of nodes V represents the physical nodes in the network (e.g. sensors in an IoT network), and the set of edges E represents the physical links between the nodes that can interact directly with each other. Mathematical functions, f, can be represented as subgraphs H of the overall network which we refer to as functional topologies. Even though the definition of functional topologies is much broader [2], in terms of INC the functional topologies are directed, rooted trees, i.e. Steiner trees, with all edges pointing towards the root, Y. This root/sink node is where the result of a particular function f must reach. The leaves of this in-tree are the inputs of the function, X. These inputs may take the form of entries in a distributed database or measurements of the environment such as temperature. The set of inputs, $x_i \in X$, are generated in nodes v_i.

The individual operations of a given function, $f(X)$, are mapped to specific nodes in the network. Each functional topology, H, is some subset of $U \subset V$ and $D \subset E$ in which each node and edge is involved in computing and routing $f(X)$. Multiple, unique functional topologies which model the same function, $f(X)$, can be chosen from the same physical topology, G. Any distinct functional topologies that perform the same calculation and result in the same observable macrostate of the network function are considered to be degenerate. The overall delay associated with a given functional topology is the principal macrostate of interest in this paper. We define the delay as the maximum graph distance between the sink node, Y, and any other node, $u \in H$. Alternative macrostates of potential interest are energy efficiency, which could be approximated by the number of edges in functional topology H, and computational throughput (number of function calculations per unit delay).

In [14], the authors made the distinction between two classes of network functions, namely divisible and indivisible. Examples of divisible functions include max, min, mean and computing the frequency histogram of sensor measurements. On the other hand, examples of indivisible functions include Fast Fourier Transform, Singular-Value Decomposition, distributed neural networks. Denote a set $W = \{1, \ldots, |W|\}$ and a subset $S = \{i_1, \ldots, i_k\} \subset W$, where $i_1, < i_2, < \ldots, < i_k$. Let x_S be a set $\{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$. A function, $f : x_S \rightarrow y$, is divisible if given any partition of S, $\Pi(S) = \{s_1, \ldots, s_j\}$ of $S \subset W$, there exists a function $g^{\Pi(S)}$ such that for any x_S,

$$f(x_S) = g^{\Pi(S)}(f(x_{s_1}), \ldots, f(x_{s_k})). \tag{1}$$

Otherwise, f is indivisible. A special case of divisible functions are so called *Symmetric Functions*, which are divisible functions that are invariant with respect to permutations (σ) of their arguments:

$$f(x_S) = f(\sigma(x_S)), \forall \sigma \tag{2}$$

A. Degeneracy of Functional Topologies

Degeneracy can be considered in terms of these two classes of network functions. We define strong degeneracy to be the multiplicity of functional topologies of an indivisible network function. We define weak degeneracy to be the multiplicity of functional topologies of divisible functions. As previously discussed in Section I, aggregate functions are of great importance for distributed computing, and since they are symmetric, we focus on the aspects of weak degeneracy in this paper.

For a physical topology, G, a network function, f, and an input & output set of nodes, X & Y, we can define *weak degeneracy* of the physical topology, G, with respect to an observable (e.g., delay, energy efficiency) as the multiplicity of functional topologies,

$$g_W(G, X, Y, f, d) = |\{H(G, X, Y, f, d)\}|, \tag{3}$$

with a given delay, d. For completeness, *strong degeneracy* is defined as the multiplicity of functional topologies on top of a physical topology G with respect to an observable, with input nodes X, performing an indivisible function in which the sub-operations must be performed in an exact ordering, $g_S(G, X, Y, f, d)$.

Since $f(X)$ is symmetric, we can perform the calculation using any partition s of X and combine the result of each subset $f(x_s)$ in any order. Weak degeneracy of a full mesh physical topology is therefore dependent on the number of ways of partitioning a set, known as the Bell number,

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_n, \tag{4}$$

for a set of $n + 1$ elements and $B_0 = B_1 = 1$. It should be noted that the Bell number is an upper bound for the weak degeneracy. The weak degeneracy of a generic physical topology has to further account for the number of mappings of these partitions to a given functional topology. This is the number of ways we can route each partition such that the suboperation $f(x_{s_i})$ on partition s_i occurs when the elements of partition s_i intersect for the first time.

Our degeneracy analysis allows us to both identify the set of subgraphs which perform a function and to sort these with regard to observables like delay, energy efficiency or computational throughput. This insight may be used in future to improve computational throughput, computational resilience, deploy superior network configurations, or to analyze the degeneracy potential of a routing protocol, i.e., how many of the feasible functional topologies can a routing protocol discover.

B. Redundancy of Functional Topologies

Degeneracy arises from structurally different functional topologies having functionally identical properties. We can also define redundancy in functional topologies, which is related to degeneracy, with two redundant functional topologies not sharing any common nodes or edges apart from their common X and Y. The redundancy between two functional topologies, $r(H_a, H_b)$, is defined as

$$r(H_a, H_b) = \begin{cases} 1, \text{ if } U_a \cap U_b = X \cup Y, \\ 0, \text{ otherwise.} \end{cases} \tag{5}$$

We define the average redundancy,

$$R(G, Y, f) = \frac{1}{|\{X\}|} \sum_{\{X\}} \frac{1}{N_X} \sum_{\substack{\{H_i, H_j\} \\ i \neq j}} r(H_i, H_j), \tag{6}$$

as the mean over the number of functional topologies which are redundant to any other given functional topology. The sum is over all N_X possible functional topologies and the set of all possible input sets, $\{X\}$. The total redundancy gives a measure of the number of ways to choose alternate functional topologies which bypass nodes in case of node/link failure.

It is to be noted that both degeneracy and redundancy directly affect the computational robustness of INC. However, while degeneracy defines the cardinality of the feasibility space for the functional embeddings, redundancy informs us about the potential of the physical topology to perform parallel computations under the same delay constraints.

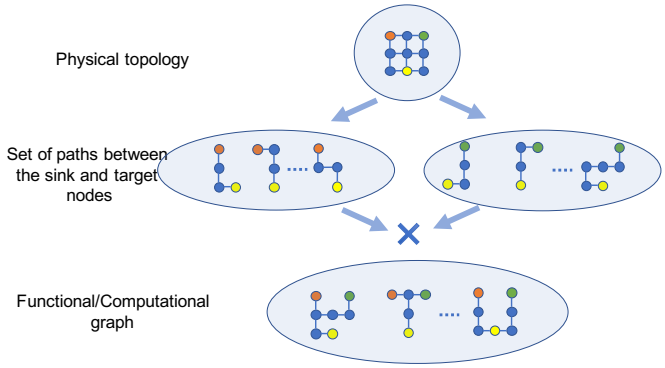

Fig. 1. Process to identify all functional topologies from a generic physical topology. The first step is to find all simple paths from the input nodes (green and orange) to the sink node (yellow). Then we take the union of those paths to generate a set of subgraphs, that is being analyzed to identify unique functional topologies.

III. Find all weakly degenerate functional topologies

In the context of INC, a functional topology describes the interactions between network nodes in the course of computing a distributed function [2]. For the INC of divisible functions functional topologies are Steiner trees, i.e. they include only the nodes and edges from the physical topology that are involved in the computation and/or routing of the function.

To generate a functional topology, consider a partition S of the input set X. For each subset $s \in S$, the function f acts on the elements in s at their first meeting point node in

Algorithm 1 Our algorithm to generate all functional topologies of a graph G, for a symmetric function f, with input nodes X, sink node, Y, and a maximum delay cutoff $d_{\max}$.

procedure FINDFTS(G, X, Y, f, $d_{\max}$)
  Find all directed, simple paths, $\{p_{x_i}\}$, from all source nodes, X, to the target node, Y of length $d_{\max}$ or less. ▷ e.g. Breadth-First Search
  Take the union of edges and nodes of all possible combinations, C, of paths from different input nodes $p_{x_i} \forall x_i$.
  if A given combination, $c \in C$, is not a tree (contains one or more cycles). **then**
    Find all unique spanning trees of c.
    for All unique spanning trees **do**
      if Spanning trees has leaf nodes, $L \nsubseteq X$. **then**
        Remove leaf nodes, L, since they are not an input.
      end if
    end for
  end if
  Remove any duplicate functional topologies.
end procedure

H. For each input set X we construct a set of paths, $P_X = \{p_1, p_2, \ldots, p_{|X|}\}$, where $p_i = (x_i, v_1, \ldots, Y)$. Let us denote by P_s the set of paths between each node $i \in s$ and Y, i.e. $P_s = \{p_i | i \in s\}$. The sub-operation $f(s)$ occurs where the paths P_s meet for the first time. For example, let us look at an input set $X = \{x_1, x_2, x_3, x_4\}$. We construct paths $P_X = \{p_1, p_2, p_3, p_4\}$, where $p_1 = (x_1, v_1, \ldots, v_l, \ldots, Y)$, $p_2 = (x_2, u_1, \ldots u_j, \ldots, Y)$, $p_3 = (x_3, w_1, \ldots, w_m, \ldots, Y)$, and $p_4 = (x_4, t_1, \ldots t_q, \ldots, Y)$. Let us consider a possible partitioning of the input set X into two partitions - $s_1 = \{x_1, x_2, x_3\}$ and $s_2 = \{x_4\}$. The sub-operation $f(s_1)$ occurs at the meeting point $v_l = u_j = w_m$. The sub-operation $f(s_1)$ is then rooted to Y following any of the original paths, i.e. $(v_l, \ldots, Y)$, $(u_j, \ldots, Y)$, or $(w_m, \ldots, Y)$. The next sub-operation $f(s_1, s_2)$ is then computed at the meeting point of any chosen original path from s_1 and the path p_4. Paths p_1, p_2, p_3, p_4 must contain at least one crossing point since they have a common root at Y. Since f is divisible, the order in which we combine paths between any input node $x \in X$ and Y does not matter.

If we consider a specific path each from x_1 and x_2 to Y and take the union of the nodes and edges in each path, we generate a subgraph of G rooted at Y and with leaves X. If the union of the paths join and later diverge, one or more undirected cycles have been formed. Each independent path around the cycle results in a different functional topology. These functional topologies can be identified by finding the unique spanning trees of the cycle-containing subgraphs and removing the resulting leaves which are not inputs, X, since they don't perform a computational or routing role.

We assume that G is connected and there exist paths from each input node to the sink. To aid in simulation, we restrict the maximum path length (delay), $d_{\max}$, to the maximum graph distance between the sink and any other node, known as the eccentricity. Figure 1 depicts this process of extracting the functional topologies from a physical topology and Algorithm 1 outlines our algorithm.

Our algorithm extends the standard Steiner Tree Packing problem and its relaxation, i.e. the Fractional Steiner Tree Problem (which are known to be NP-hard [18]), by extending the search from minimum weight Steiner trees to all existing Steiner trees in which the sum of the weights on the links is lower than a predefined value. To the best of our knowledge, this problem has not been addressed in the literature so far.

A. Complexity Analysis

In Algorithm 1, the most computationally expensive task is the search algorithm that finds all directed, simple paths p_{x_i}. For example, in general the time complexity of the Breadth-First Search algorithm can be expressed as $O(|V| + |E|)$. However, Algorithm 1 defines the maximum path length d_{max}, which reduces the search space, resulting in a time complexity that can be expressed as $O\left(b^{d_{max}}\right)$, where b is the branching factor (i.e. the average out-degree) [25]. The steps after the search algorithm include:

- combining the discovered paths - the time complexity of the union operation is $O(n)$, where n is the number of elements/nodes in the set. In our case, due to the upper bound in terms of path length, the max number of elements in the edge set is $|s|\, d_{max}$. Hence the time complexity of the union operation is $O\left(|s|\, d_{max}\right)$.
- finding the spanning trees - the time complexity of the Kruskal algorithm [26] to find spanning trees is $O\left(m \log\left(n\right)\right)$, where m is the number of edges and n is the number of nodes in the graph. Due to our limit in terms of d_{max} the time complexity can be expressed as $O(|s|\, d_{max} \log\left(|s|\, d_{max}\right))$

The time complexity of Algorithm 1 can be calculated as:

$$T(n) = b^{d_{max}} + |s|\, d_{max} + |s|\, d_{max} \log\left(|s|\, d_{max}\right) \tag{7}$$

The time complexity of the search algorithm is obviously the dominant one, meaning that the complexity of the whole algorithm can be expressed as:

$$T(n) = O\left(b^{d_{max}}\right) \tag{8}$$

The maximum path length between nodes in a sensor network (d_{max}) is a design parameter that is usually set to a small value to ensure low latency and high energy efficiency. The network topology is usually planned in a way that keeps this value small even for a growing network size. An example of a network that is constantly growing without affecting the average path length between the nodes is the Internet (its topology is believed to exhibit a power low distribution [27]). Therefore, Algorithm 1 is efficient enough to be used in a realistic network scenario.

IV. Analysis

We now present the numerical analysis of the degeneracy and redundancy of three different physical topologies: (1) an

11×11 square lattice; (2) a randomly placed sensor network with the same number of nodes; and (3) a real-world sensor network deployed at the Intel Research Lab in Berkeley [28]. In the lattice case, we place the sink node, Y, in the centre of the lattice. In the case of a random topology, we consider an area of 1×1 km, place the sink node at the centre of the area, and randomly distribute the remaining 120 nodes. The edges are chosen to connect each node to its four closest neighbors[2]. In the case of the real-world sensor network deployed at the Intel Research Lab, the network consists of 54 Mica2Dot sensors, which collected data for about 5 weeks. We used the averaged connectivity data, consisting of sender id, receiver id, and probability of a message from a sender successfully reaching a receiver, to compute the physical topology. The simulation involves calculating all functional topologies for randomly sampled input node pairs, X, which are chosen to be within three hops from each other, since in-network computation is typically used for data collected by nearby sensors. To calculate the average number of degenerate functional topologies with a given delay, this process is repeated 500 times. The standard errors on the means are estimated using bootstrap resampling of all simulations, and are in-fact smaller than the plot markers.

The cumulative degeneracy of functional topologies and cumulative average redundancy for the lattice and random topologies are shown in Figure 2, while the same results for the network deployed at the Intel Research Lab are shown in Figure 3. For all three topologies the maximum delay considered is equal to the eccentricity of the sink node, which is 10 in the case of the lattice and random topology, and 6 for the Intel dataset. For the lattice case, the cumulative number of degenerate functional topologies (blue markers) is seen to increase exponentially with the increasing delay limit. This same exponential behaviour has also been seen in independent experiments on different lattice sizes. It is a result of the branching process associated with functional topologies of increasing size gaining access to more and more nodes and their respective edges. The cumulative redundancy of functional topologies (red markers) also increases exponentially with delay limit. Although the cumulative degeneracy and redundancy are lower in the case of a random topology than those observed in a lattice, they still increase exponentially as the delay limit increases. As can be observed by comparing Figure 2 and Figure 3, the degeneracy values of the Intel deployment are similar to the square lattice deployment, whereas the redundancy values are lower compared to the square lattice and higher as compared to the random network deployment. The increase in cumulative redundancy demonstrates that the ability for physical topologies to perform parallel computing using independent functional topologies is higher for less restrictive delay limits.

Using the numerical estimates of degeneracy, we now examine the computational robustness of INC. Let us first define the probability of computational success, α_i, at node i as the number of successful computations as a fraction of

[2]It is worth noting that this results in a non-uniform distribution of the edges.

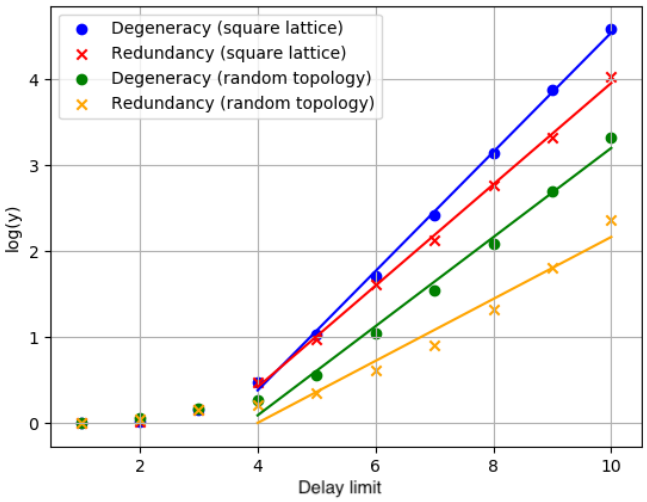

Fig. 2. Log plot (base 10) of cumulative degeneracy and cumulative redundancy. For the square lattice, the slopes of the linear fits for the cumulative degeneracy and cumulative redundancy are: 0.69 and 0.59, respectively. For the random topology, the slopes of the linear fits for the cumulative degeneracy and cumulative redundancy are 0.51 and 0.36, respectively.

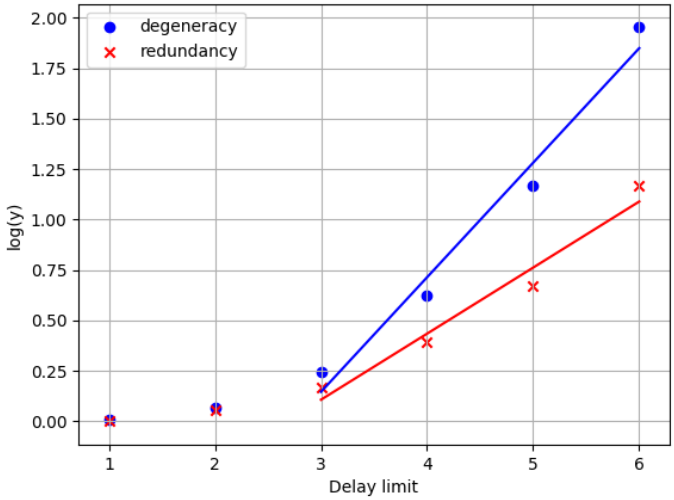

Fig. 3. Log plot (base 10) of cumulative degeneracy and cumulative redundancy for the Intel dataset. The linear fits for the cumulative degeneracy and cumulative redundancy are: 0.66 and 0.39 respectively.

the total number of attempts. The probability of a node failing during the computation of a function is then found to be,

$$P_{f_i} = P_{c_i} \cdot (1 - \alpha_i), \tag{9}$$

where, P_{c_i}, is the probability of a node occurring in any feasible functional topology. A computation may fail at a node for a number of reasons such as excessive load or during a sleep cycle. The probability of a successful computation is given as

$$P_{S_c} = \prod_{i \in U} (1 - P_{f_i}), \tag{10}$$

where U are the nodes in the functional topology that performs the computation. Figure 4 compares the probability of successful computation using a single functional topology, e.g. selected as the optimal computational graph in terms

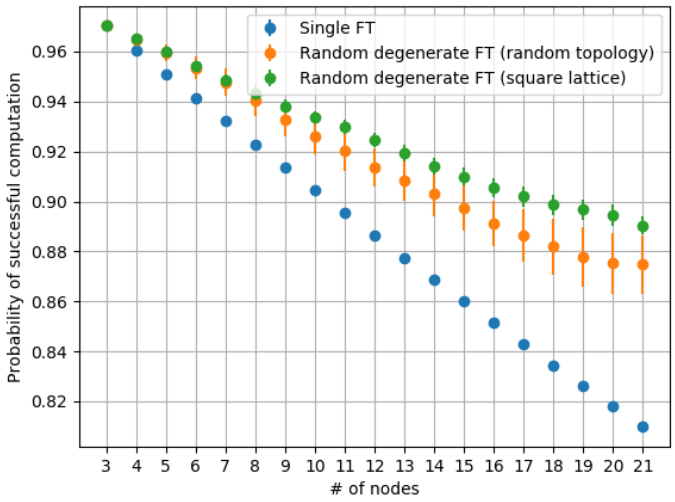

Fig. 4. Probability of successful computation vs number of nodes involved in the computation. The blue dots refer to a single functional topology; the green and orange dots refer to a randomly selected functional topology that can perform the same function for the square lattice and random topology respectively. The delay limit for all calculations is set to be equal to 10.

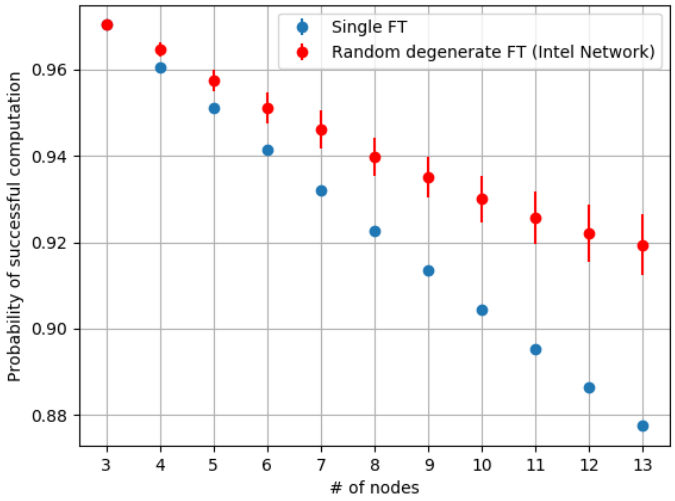

Fig. 5. Probability of successful computation vs number of nodes involved in the computation for the Intel Network. The blue dots refer to a single functional topology; the red dots refer to a randomly selected functional topology that can perform the same function for the Intel Network. The delay limit for all calculations is set to be equal to 6.

of delay or energy consumption, and a random selection of one of the multiple functional topologies that can perform the same computation while satisfying a maximum delay requirement (10 for the results in figure) for the lattice and random topology. Figure 5 shows the same results for the Intel network deployment. In this case the delay limit is 6. In the case of a single optimal functional topology, $P_{c_i} = 1$ for all nodes, since all nodes in the optimal graph must be used for INC. Considering the degeneracy of functional topologies from Figure 2 and Figure 3, each computation can potentially be performed by using any of the degenerate functional topologies. Therefore, we estimate P_{c_i} for the randomly selected functional topology, that is going to perform the computation, as the frequency of occurrences of the node

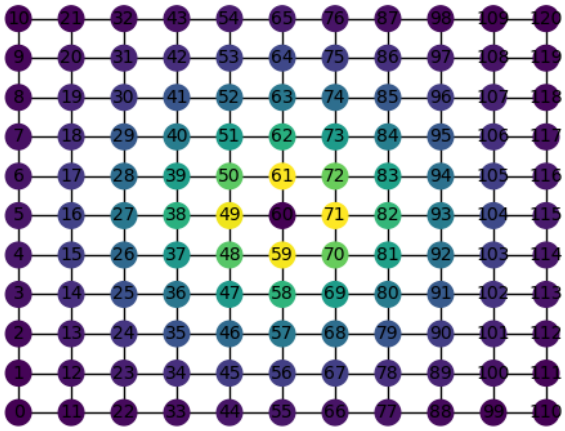

Fig. 6. Heat-map of the square lattice topology showing the probability of a node occurring in any feasible functional topology. Brighter colors represent higher probabilities. The sink node is in the center of the square lattice and the input nodes are randomly sampled and within three hops from each other.

i in all the degenerate functional topologies. This P_{c_i} is then used to compute the corresponding P_{S_c} according to (9) and (10).

Results are then averaged over all the 500 input pairs. The results in Figure 4 and Figure 5 show that it is possible to significantly increase the probability of successful computation by exploiting the multiplicity of functional topologies, i.e. the degeneracy.

The previous results show that the awareness of computational alternatives allows us to increase the probability of successful computations. We will take a closer look at the square lattice topology to understand how the structure of the physical topology affects the degeneracy and redundancy in terms of INC. Figure 6 shows the heat-map of the square lattice topology representing the probability of each node being included in a functional topology. Brighter colors correspond to higher probabilities. Evidently, Figure 6 shows that the nodes closer to the sink are being included in more computations, i.e. more functional topologies. Therefore, the number of nodes connected to the sink node directly affects the number of ways to reach the sink node within a limited number of hops, which is defined by d_{max}. These nodes are also the communication bottlenecks of the physical topology. Additionally, Figure 6 shows that for example node 72 has a higher probability to be included in a functional topology than node 82. This is interesting because both nodes (72 and 82) are two hops away from the sink node. However, the difference arises from their positions and the connectivity pattern in the physical topology: due to the Von Neumann neighborhood connectivity, node 72 is located so that it provides shortest path connectivity for more nodes (i.e. the nodes in the upper and right part of the graph) compared to node 82 (i.e. the nodes in the right part of the graph).

The regularity of the square lattice results in a fairly uniform probability of nodes being included in the functional topologies (see Figure 6). However, things are quite different for topologies that are not so regular, e.g. the random topology or the Intel deployment. This non-regularity explains the

high variance in Figure 4 and Figure 5, and the larger gap between degeneracy and redundancy of the random topology compared to the square lattice shown in Figure 2, and between degeneracy and redundancy of the Intel topology in Figure 3.

V. Conclusion

We presented a novel way of considering INC using ideas from statistical mechanics. In particular, we provide a formal definition and calculation of degeneracy and the related concept of redundancy for distributed computations in a network. We also introduce an algorithm to efficiently compute all degenerate and redundant functional topologies.

The time complexity of the proposed algorithm is exponential with the increasing path length in the network. However, the maximum path length is a design parameter which is usually kept small for a growing network size, allowing us to use the proposed approach to analyze degeneracy and redundancy of realistic network topologies.

Our results show that the cumulative degeneracy and redundancy of functional topologies increase exponentially as the accepted delay limit for the computation is increased. The analysis of different physical topologies shows that the underlying network structure affects the operation of network functions and results in different values of degeneracy and redundancy. The successful computation rate of a symmetric function is shown to be significantly higher using the set of possible degenerate functional topologies as opposed to exclusively using the optimal functional topology. This means that we can considerably improve the robustness of the computation, while still being able to satisfy requirements, such as delay or energy consumption. Future work will focus on exploiting degenerate and redundant functional topologies to perform parallel computing. The case of redundant, i.e. independent functional topologies, is of particular interest in that bottlenecks can be avoided with no coordination between the nodes of different functional topologies when performing the same network function. We will investigate mechanisms to balance nodes usage for redundant topologies implementing multiple coexisting functions across the same physical network. The same mechanisms will be exploited for the case of degenerate topologies.

Acknowledgment

This publication has emanated from research supported in part by a research grant from Science Foundation Ireland (SFI) and is co-funded under the European Regional Development Fund under Grant Number 13/RC/2077, in part by work supported by the Air Force Office of Scientific Research under award number FA9550-17-1-0066, and in part by CHIST-ERA (call 2017) via the FIREMAN consortium, which is partly funded by the Irish Research Council.

References

[1] P. H. Mason, "Degeneracy: Demystifying and destigmatizing a core concept in systems biology," *Complexity*, vol. 20, no. 3, 2015.

[2] M. Dzaferagic *et al.*, "A functional complexity framework for the analysis of telecommunication networks," *Journal of Complex Networks*, 2018.

[3] V. Shah, B. K. Dey, and D. Manjunath, "Network flows for function computation," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 4, pp. 714–730, 2013.

[4] S. Kannan and P. Viswanath, "Multi-session function computation and multicasting in undirected graphs," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 4, pp. 702–713, 2013.

[5] J. Liu, C. H. Xia, N. B. Shroff, and X. Zhang, "On distributed computation rate optimization for deploying cloud computing programming frameworks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 4, pp. 63–72, 2013.

[6] P. Vyavahare *et al.*, "Optimal embedding of functions for in-network computation: Complexity analysis and algorithms," *IEEE/ACM Transactions on Networking (TON)*, vol. 24, no. 4, pp. 2019–2032, 2016.

[7] Y. Yang, S. Kar, and P. Grover, "Graph codes for distributed instant message collection in an arbitrary noisy broadcast network," *IEEE Transactions on Information Theory*, vol. 63, no. 9, 2017.

[8] E. Di Pascale, I. Macaluso, A. Nag, M. Kelly, and L. Doyle, "The Network as a Computer: a Framework for Distributed Computing over IoT Mesh Networks," *IEEE Internet of Things Journal*, 2018.

[9] A. Giridhar and P. R. Kumar, "Toward a theory of in-network computation in wireless sensor networks," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 98–107, 2006.

[10] A. Jindal and M. Liu, "Networked computing in wireless sensor networks for structural health monitoring," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 4, pp. 1203–1216, 2012.

[11] F. Zeng, C. Li, and Z. Tian, "Distributed compressive spectrum sensing in cooperative multihop cognitive networks," *IEEE Journal on Selected Topics in Signal Processing*, vol. 5, no. 1, pp. 37–48, 2011.

[12] R. Olfati-Saber, "Distributed kalman filtering for sensor networks," *IEEE Conference on Decision and Control*, pp. 5492–5498, Dec 2007.

[13] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: an acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 122–173, 2005.

[14] A. Giridhar and P. R. Kumar, "Computing and Communicating Functions Over Sensor Networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 755–764, 2005.

[15] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "in-Network Aggregation Techniques for Wireless Sensor Networks: a Survey," *IEEE Wireless Communication*, no. April, pp. 70–87, 2007.

[16] S. Madden, R. Szewczyk, M. J. Franklin, and D. Culler, "Supporting aggregate queries over ad-hoc wireless sensor networks," *Proceedings - 4th IEEE Workshop on Mobile Computing Systems and Applications, WMCSA 2002*, pp. 49–58, 2002.

[17] O. Diallo, J. J. P. C. Rodrigues, M. Sene, and J. Lloret, "Distributed database management techniques for wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 2, pp. 604–620, 2015.

[18] J. Cheriyan and M. R. Salavatipour, "Packing element-disjoint steiner trees," *ACM Transactions on Algorithms (TALG)*, vol. 3, no. 4, 2007.

[19] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[20] A. Haider, R. Potter, and A. Nakao, "Challenges in resource allocation in network virtualization," in *20th ITC Specialist Seminar*, vol. 18, no. 2009. ITC, 2009.

[21] A. Belbekkouche, M. M. Hasan, and A. Karmouch, "Resource discovery and allocation in network virtualization," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 1114–1128, 2012.

[22] J. Inführ and G. R. Raidl, "Introducing the virtual network mapping problem with delay, routing and location constraints," in *International Conference on Network Optimization*. Springer, 2011, pp. 105–117.

[23] W.-L. Yeow, C. Westphal, and U. Kozat, "Designing and embedding reliable virtual infrastructures," in *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*. ACM, 2010, pp. 33–40.

[24] J. F. Botero, X. Hesselbach, A. Fischer, and H. De Meer, "Optimal mapping of virtual networks with hidden hops," *Telecommunication Systems*, vol. 51, no. 4, pp. 273–282, 2012.

[25] R. E. Korf, "Depth-First Iterative-Deepening: An Optimal Admissible Tree Search," *Artificial Intelligence*, vol. 27, no. 1, pp. 97–109, 1985.

[26] J. B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," *Proceedings of the American Mathematical society*, vol. 7, no. 1, pp. 48–50, 1956.

[27] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On Power-Law Relationships of the Internet Topology," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 4, pp. 251–262, 1999.

[28] C. G. S. M. M. P. Peter Bodik, Wei Hong and R. Thibaux, "Intel Berkeley Research Lab Data (v. 2004-04-28)," http://db.csail.mit.edu/labdata/labdata.html, Apr. 2004.