Virtual Network Embedding for Wireless Sensor Networks

Time Efficient QoS/QoI Aware Approach

Roland Katona, Victor Cionca, Donna O'Shea, Dirk Pesch

***Abstract*—A recent trend in Wireless Sensor Networks (WSN) is Network Virtualization to support on-demand sharing of sensing functionality. The efficient allocation of WSN resources to sensing requests is obtained using Virtual Network Embedding (VNE). This must take into account Quality of Service - QoS (e.g. reliability), Quality of Information - QoI (e.g sensing accuracy), and deal with wireless interference. With increased computational complexity due to the added constraints, finding an optimal solution can be prohibitive at scale. We developed an offline embedding algorithm that searches through all possible embeddings, which allowed us to *explore the trade-off between solution quality and search time*. We identify a defined set of initial processing steps that lead to high quality solutions (within 10% of best solution) in bounded time. We evaluated the algorithm under high stress (large networks with long paths, high data rates, beyond typical WSN configuration) to understand its limitations and the limitations imposed by the underlying WSN substrate.**

***Index Terms*—virtual network embedding, wireless sensor networks, quality of service, quality of information, resource allocation, resource management**

I. Introduction

Following its success in enterprise networks, Network Virtualization (NV) [1] has become an attractive topic of research in Wireless Sensor Networks (WSNs). Virtualised Wireless Sensor Networks (VWSN) have been proposed to overcome the inefficiencies and limitations of traditional, proprietary, single purpose, single user WSNs. By virtualising the physical node (e.g. sensing and processing) and link (e.g. radio channel) resources of WSNs and by opening up existing WSNs as shared, multi user sensing infrastructures, these conventionally closed deployments can be employed beyond the scope of their original function, allowing them to evolve and support the on-demand provisioning of multiple co-existing virtual networks, each with its own applications and custom performance levels.

There are two challenges in NV: partitioning the physical network to isolate virtual networks; and finding the optimal mapping of physical resources to virtual networks. This paper focuses on the second, known as Virtual Network Embedding (VNE), and proposes a solution for WSNs. In VWSN, users submit Virtual Network Requests to deploy sensing applications over the shared WSN substrate. The VNE must identify an allocation of physical node (e.g. sensing and processing) and link (e.g. wireless channel) resources to Virtual Network Requests (VNR) that optimizes an objective (e.g. cost or profit), while respecting the substrate's capacity limitations and meeting the specific service and performance requirements of the request.

The VNE problem in wireless sensor networks (and wireless mesh networks in general) is a relatively new undertaking compared to its counterpart in wired networks. There are many exact [2] and heuristic [3] solutions for the latter. However the VNE in WSN is significantly different from its wired counterpart because of the characteristics of the applications and of the physical substrate. Dealing with these differences brings specific constraints into the VNE and increases the complexity. First, WSN applications are data and location driven so applications will request sensing at specific locations with bounded measurement error (*Quality of Information* – QoI) which constrains the mapping of virtual to physical nodes. Second, the data must be reported reliably and within a certain time bound (*Quality of Service* – QoS), which constrains the allocation of communication resources (physical paths). The allocation of wireless link resources is further complicated by the broadcast nature of the wireless channel, as an on-going transmission on one link interferes with other neighboring links. Therefore, the embedding must take the impact of link activation on the entire interference neighbourhood into account. Another difference is the reduced energy levels available in WSN: the VNE must ensure uniform energy consumption to maximize network lifetime.

The VNE problem is known to be NP-Hard [4]. Finding an optimal solution for large networks with many VNRs will require prohibitive processing time. In these situations it is better to identify a heuristic that provides a "*quick & good*" solution. In this paper we explore the trade-off between the processing time (*quick*) and solution quality (*good*). We developed an algorithm that explores the entire space of possible embedding solutions using greedy heuristics and combinatorial optimization. The algorithm is for offline embedding (embeds batches of VNRs at once) and takes into account the constraints on QoS (in terms of end-to-end reliability), QoI (in terms of sensing accuracy), as well as fine-grained inter-link interference. The algorithm can be interrupted at any point in the search, returning the *current best* solution. By analysing

Roland Katona, Victor Cionca, and Donna O'Shea are with the Computer Science Dept., Cork Institute of Technology, Ireland, email: {firstname.lastname}@cit.ie.

Dirk Pesch is with the School of Computer Science and IT, University College Cork, Ireland, email: d.pesch@cs.ucc.ie.

The work was funded by Science Foundation Ireland (SFI) under grant 13/IA/1885.

the variation of the solution quality versus the search time we identified that *a defined set of initial processing steps is enough to achieve >90% of the best quality in 90% of cases*. Since this initial processing is bounded in time it can be used as a heuristic for obtaining the desired *quick & good* solutions. The performance of the algorithm was compared with that of a Mixed Integer Linear Programming (MILP) solver. We stressed the parameters of the problem (data rate, network diameter) beyond the common configuration of WSNs in order to obtain a general understanding of the behaviour and limitations of VNE in WSN. Our contributions are:

- exploration of the trade-off between solution quality and processing time
- identification of a heuristic that obtains high quality solutions in bounded time
- analysis of the limitations of VNE in WSN at scale (large networks, high data rate).

In the following we first provide a formal definition of the VNE problem in WSN (II). The VNE algorithm is discussed in great detail, explaining the heuristics and techniques employed (III). An evaluation baseline, the experimental setup, as well as the experiment results are presented in (IV). We further discuss the results and other insights from the paper in (V). Before concluding, related work is presented in (VI).

II. VNE Problem Definition and System Model

We assume a virtualized WSN with a single sink where, for simplicity, all the nodes measure the same physical process (*e.g.* temperature). Initially the WSN does not run any applications. Users submit requests for deploying sensing applications on the WSN, specifying the sensing target locations, the tolerated measurement error (QoI), the data rate, and minimum reliability (QoS). This is a Virtual Network Request (VNR) and it is processed by a WSN controller, that can be co-located with the WSN gateway. The controller collects and maintains the network state [5] and runs the VNE algorithm on the current network state to embed batches of VNRs at once. After embedding, the network state is updated to reflect the new allocations, and the user applications are deployed on the reserved resources (this step is not considered in this paper). The network state consists of WSN topology with node locations, logical connectivity and link quality (*i.e.* reliability); and available node and link resources [6]. According to the taxonomy in [4], the VNE algorithm is *static* (requests are known in advance) and *centralized* (solved on WSN controller). It considers the following constraints:

- the QoS and QoI requirements of the VNRs
- the limited capacity of the wireless links
- capacity degradation caused by transmission collisions.

The substrate network is represented as a directed, weighted and capacitated graph $G = (N, E)$. For each node $n_i \in N$ denote $p_i(x, y)$ its location on a geographical plane. On the same plane define a uniform grid of locations S, the Sampling Request Points (SRP). Users making requests (VNRs) for data collection will refer to an $s_i(x, y) \in S$ to indicate the location where sampling is needed. Values measured at an SRP will differ from sensors around it, with the differences assumed known in the error matrix D, where $D_{ij} = \Theta(p_i) - \Theta(s_j)$, and $\Theta(p_i)$ represents the expected value of the physical process at coordinate p_i.

For each WSN link $e_i \in E$ that has exchanged at least one packet, define packet success rate (reliability) e_i^{μ}, load e_i^{φ} and capacity e_i^{Z}, all normalized to be within $[0, 100]$. The packet success rate includes MAC retransmissions. The link load is the proportion of link capacity *currently* being consumed and the capacity is the maximum amount of data that can be sent over the link for a given unit of time. Furthermore, we define a link *weight* property that combines link load and reliability as per Eq. 1, where c_{φ} and c_{μ} are coefficients.

$$e_i^{\psi} = c_{\varphi} \cdot e_i^{\varphi} + c_{\mu} \cdot (100 - e_i^{\mu}) \tag{1}$$

Virtual Network Requests are defined as $R \in \mathcal{R}, R_i = \{R_i^s, R_i^{\delta}, R_i^{\pi}, R_i^{\gamma}\}$, where $R_i^s < |S|$ is the index of an SRP where measurements are requested, R_i^{δ} is the QoI in terms of tolerated measurement error, R_i^{π} is the QoS (*i.e.* reliability), and R_i^{γ} is the requested sampling rate (samples per second). The requested sampling rate is converted to link capacity usage *quota* using the known application payload size (σ) and the maximum link data rate (ρ, obtained empirically), as $(\sigma R_i^{\gamma})/\rho$. Henceforth R_i^{γ} will denote the VNR requested quota.

The VNE algorithm must find, for each VNR, a source node that satisfies the tolerated error R_i^{δ}. The destination for all VNRs is the sink, so the VNE must find the path between source and sink that satisfies the reliability R_i^{π} and can support the quota R_i^{γ}. The objective is to embed as many VNRs as possible while consuming the least network resources.

Wireless interference that causes colliding transmissions must be prevented, otherwise the reliability constraint will be violated. To achieve this the VNE allocates link capacity (*e.g.* transmission slots) such that when a link is active all the other links in its Interference Neighborhood (IN) are inactive. This is a *sufficient condition* for conflict-free communication [7]. The IN of link e_i is calculated using the protocol model [8] based on node connectivity information available in the network state. When calculating the existing load on a link (ϵ^{φ}), in addition to what was allocated on it directly, we will also take into account the indirect quota on all the links in its IN. Even though the link will not transmit during the indirect time quota, it must remain silent to prevent collisions with its neighbors. As such we denote by $I_i = IN(e_i) \bigcup \{e_i\}$ the set of links affected by quota allocation on link e_i.

The mapping of a VNR to the WSN substrate is represented formally with two binary variables: $x_{ij} \in \{0, 1\}$ for mapping of VNR R_i to WSN node n_j; and $y_{ij} \in \{0, 1\}$ for allocating link e_j to R_i. The VNE problem can then be expressed as the Mixed Integer Linear Program (MILP) represented in equations 2-7. It is a bi-level objective problem, the primary objective (Eq. 2) to maximize admission rate (number of mapped VNRs) and the secondary to minimize cost (Eq. 3). The cost is defined as the total load allocated on all network links, including the impact of interference. Eq. 4 models the link capacity constraint that takes into account the entire link IN. Eq. 5 restricts VNR source nodes to WSN nodes with acceptable measurement error from the SRP. Path reliability is the product of the reliability of links where R_i was embedded,

which can be represented as $\prod(e_j^{\mu})^{y_{ij}}$ [1], and this needs to be $\geq R_i^{\pi}$, the required reliability of the VNR. By applying the logarithm on both sides of the constraint we obtain Eq. 6. Eq. 7 ensures that each VNR is only mapped to one source if it is accepted. Eq. 8 represents flow conservation for *regular nodes*, where the outbound data is equal to the sum of the inbound data and the generated data. Eq 9 is flow conservation for the network sink (N_S); the sink does not generate data and there is no outbound data either, so the total inbound data is equal to the data generated by all mapped VNRs. Eq. 10 and Eq. 11 constrain the traffic from a VNR on a single path (no fractional flows). In Eq. 8 to 11 $e_i \rightarrow n_k$ represents inbound links on node n_k, and $e_i \leftarrow n_k$ outbound links.

$$\max \sum_{i \leq |\mathcal{R}|, j \leq |N|} x_{ij} \tag{2}$$

$$\min \sum_{e_i \in E} \sum_{e_j \in I_i} \sum_{R_k \in \mathcal{R}} y_{kj} R_k^{\gamma} \tag{3}$$

$s.t.$

$$\sum_{e_k \in I_j} \sum_{R_i \in \mathcal{R}} y_{ik} R_i^{\gamma} \leq e_j^{Z}, \forall e_j \in E \tag{4}$$

$$x_{ij} D_{iR_i^s} < R_i^{\delta}, \forall i \leq |\mathcal{R}|, j \leq |N| \tag{5}$$

$$\sum_{e_j \in E} y_{ij} ln(e_j^{\mu}) >= ln(R_i^{\pi}), \forall R_i \in \mathcal{R} \tag{6}$$

$$\sum_{j \leq |N|} x_{ij} <= 1, \forall R_i \in \mathcal{R} \tag{7}$$

$$\sum_{e_i \rightarrow n_k} \sum_{R_i \in \mathcal{R}} y_{ij} R_i^{\gamma} + \sum_{R_i \in \mathcal{R}} x_{ik} R_i^{\gamma} = \sum_{e_i \leftarrow n_k} \sum_{R_i \in \mathcal{R}} y_{ij} R_i^{\gamma}, \forall n_k \in N - \{N_S\} \tag{8}$$

$$\sum_{e_i \rightarrow N_S} \sum_{R_i \in \mathcal{R}} y_{ij} R_i^{\gamma} = \sum_{j \leq |N|} x_{ij} R_i^{\gamma} \tag{9}$$

$$\sum_{e_j \rightarrow n} y_{ij} \leq 1, \forall n \in N, \forall R_i \in \mathcal{R} \tag{10}$$

$$\sum_{e_j \leftarrow n} y_{ij} \leq 1, \forall n \in N - \{N_S\}, \forall R_i \in \mathcal{R} \tag{11}$$

III. VNE Algorithm for VWSN

We introduce a novel VNE algorithm that considers the sensing accuracy of the nodes (i.e. QoI), the reliability of the communication paths (i.e. QoS) and the impact of multi-access link interference on the overall consumed communication resources. The designed algorithm processes batches of input VNRs. The batch is first sorted in non-decreasing order of requested quota, then the VNRs are processed sequentially, embedding each VNR and then updating the substrate state to reflect the allocation of the VNR resources. Single VNRs are embedded in a greedy fashion, always selecting the best possible substrate resources. Given that the order of embedding the VNRs in a list influences the quality of the result, the algorithm explores the entire set of permutations of the input VNRs to find the embedding sequence that yields the *highest acceptance ratio* with the *lowest cost* (further discussed in Sec. III-D).

The node mapping and link mapping are decoupled: first the source nodes are mapped for all VNRs at once (further discussed in Sec. III-A); then, the link mapping is performed per VNR (further discussed in Sec. III-C), given the mapped sources and following the order of the input batch. The search space and the processing time are reduced using pruning and memoization techniques, and also by taking advantage of the parallelizable nature of the combinatorial search which allows multiple concurrent processes to perform computations independently.

A. Node mapping

The algorithm starts by identifying the *source* nodes that will be allocated for the requests in the batch of VNRs (Alg. 1. line 1.). Each VNR contains an SRP (Sampling Request Point) that specifies the location where sensing is requested. The VNR further constrains the location by specifying a tolerated measurement error (Eq. 5). The search for feasible source nodes for a VNR is limited to the 2-hop neighborhood of the substrate node closest to the SRP. This threshold was chosen empirically following test results that showed that as the distance from the SRP increases the probability of satisfying the tolerated measurement error decreases, and is also supported by the literature [9]. VNRs that don't have suitable candidates are rejected at this point and not included in the link mapping step.

When several VNRs share the same physical node, they will have a higher probability of contending with each other for link resources. This is mitigated by distributing the VNRs over as many candidate substrate nodes as possible, in order to reduce the inter-flow interference among VNRs. The distance (in terms of hop count) of candidate source nodes to the sink is also taken into account, giving preference to those closer to the sink, which tends to result in shorter substrate paths and better end-to-end reliability.

B. Determining bounds on the acceptance ratio

In this paper the WSN is assumed to have a single sink, and all the VNR traffic is assumed to be extracted through the sink. As such, the sink becomes a known bottleneck in the network. Hence, if the total VNR demanded quota is greater than the capacity of the sink then some VNRs will not be accepted. In this step, the algorithm determines the maximum acceptance ratio of the input VNRs that can be supported at the sink (Alg. 1. line 4.).

When calculating the total quota demanded by VNRs at the sink one must take into account the impact of Intra-Flow Interference (IFI). Due to IFI, links that are adjacent to an active link must be silenced to avoid collision which increases the overall required quota to be allocated. Depending on the length of a VNRs mapped path, a link might suffer from IFI multiple consecutive time slots, according to [7]. Therefore, to prevent the interference and ensure conflict-free transmission,

[1]The representation works as follows. For a given VNR R_i, the physical links allocated will have $y_{ij} = 1$ and therefore a factor equal to their reliability, e_j^{μ}. Links not used will have $y_{ij} = 0$ so a factor of 1 that does not affect the product.

the total quota allocated on the link must be multiplied by the number of slots with IFI.

Based on the shortest hop path between a VNR's mapped source node and the sink, the lowest IFI multiplier can be determined as per [7]. Applying the IFI multiplier to the requested quota, the actual required quota at the sink can be determined for each VNR. Then using this to calculate the total quota required for a set of VNRs at the sink, the algorithm will identify the largest combinations of VNRs that do not exceed the available capacity of the sink. These are obtained by solving a 0-1 Knapsack problem [10] where the element weights are the lowest required quotas at the sink. The length of the Knapsack solution represents the highest number of VNRs that can be accepted and an upper bound on the acceptance ratio. The upper bound is then used by the algorithm to filter out embedding solutions that yield a lower acceptance ratio, thus avoiding unnecessary computation (further discussed in Sec. III-D).

C. Link mapping

Once the source nodes are identified in the node mapping step, the link mapping will find a path between the source node and the WSN sink for each VNR (Alg. 1. line 11.). The path must have the lowest cost, must satisfy the reliability constraint (Eq. 6), and the links on the path as well as in their Interference Neighborhood (IN) must have sufficient capacity available to accommodate the additional quota requested by the VNR. This is a lowest cost path problem with additional constraints, known to be NP-Complete [11]. An extension to Dijkstra's path finding algorithm was developed as follows. It starts by identifying the least cost path using the link metric e^{ψ} (Sec. II). Then, based on the definition of e^{ψ} the least cost path will be the one that has the most available capacity and that is most reliable (compared to paths with similar capacity and length). The path selection does not enforce constraints, so after finding the least cost path its feasibility is validated against the reliability and capacity constraints (Alg. 1. line 16.). In case there is a violation of the above constraints, the links that caused the violation are penalized by increasing their weight with a penalty value that is larger than the sum of all network link weights. If the reliability check fails, the worst link is penalized; if the capacity check fails, the link with highest capacity overflow is penalized. Then, considering the updated link weights, a new path is sought which will avoid penalized links if possible. The process repeats until either a feasible path is found or the obtained path cost is higher than the penalization value, which indicates that the only way to the sink is through already penalized, i.e. non-feasible links, so the VNR is rejected.

D. Combinatorial search

As discussed previously, the link mapping is performed in a greedy fashion for each request. This causes the quality of the embedding (i.e. acceptance ratio and cost) to depend on the embedding sequence. To find the best embedding sequence, the algorithm analyzes the different permutations of the set of input VNRs (Alg. 1. line 7.). First, the list of feasible VNRs (i.e. the ones that were successfully mapped in the node mapping step) is sorted in non-decreasing order based on their requested quota. By starting with the lower quota VNRs we can embed more VNRs than if we started with the higher quota ones. Then, the permutations are generated in lexicographic order, divided into p contiguous blocks and each block is allocated to a different process [2].

Searching through all the permutations of the input VNRs has a factorial time complexity, however this can be greatly reduced by exploiting the following characteristics: *i)* the embedding of permutations of VNR sequences consists of dependent overlapping sub-problems, as permutations share sub-sequences (*e.g. ABC* is a sub-sequence to *ABCED*); *ii)* it also has an optimal substructure, since solutions to longer sequences can be constructed by using the solutions of sub-problems (i.e. sub-sequences). These characteristics allow the use of *Dynamic Programming (DP)* techniques such as *memoization* to reduce the processing time.

Memoization refers to caching the results of computationally intensive tasks so that they are readily available upon re-execution of the tasks, achieving a significant reduction in the amount of redundant calculations at the expense of storage space. In the VNE case, memoization was employed by our algorithm to store the embedding outcome (success or failure) of sub-sequences of one or more VNRs as well as the state of the substrate after the embedding. The state of the substrate is represented as the sub-sequence of VNRs already embedded on the substrate. For example, consider the sequence $ABCD$, where AB were accepted, C rejected, and D accepted. The algorithm will memoize the failure of C over the substrate state AB as well as the success of D over the substrate state AB (since C failed, the substrate state did not change). The following types of search pruning can be accomplished thanks to the memoization of intermediate embedding results:

- Once a permutation sub-sequence is processed, its resulting network state can be reused in any other permutation where the same sub-sequence occurs as a prefix. For example, with the batch $ABCDEF$, once $ABCD$ is evaluated AB can be reused for $ABDC$, $ABDE$, ABF, $ABEF$, etc.
- if a VNR fails on a substrate state it will also fail if more successful embeddings are added to the substrate; e.g. if C fails in $ABCD$ it will also fail in $ABDC$. Furthermore, if both C and D fail in $ABCD$ they will also fail in $ABDC$, because the substrate state for D in $ABCD$ is still AB due to the failure of C.

Since permutations are processed in an ordered manner, the memoized results are stored only as long as they serve as a dependency for later sequences. For example, using the above representation of the requests, sub-sequence AB shares a prefix with $ABCD$ and $ABDC$ but it does not with $DBCA$, and so cannot be re-used anymore. Therefore, it can be discarded to reduce the space complexity associated with caching.

To further reduce the processing time additional pruning is performed during the search for the highest quality embedding

[2]For brevity, this is not shown in Alg. 1.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3009834, IEEE Internet of Things Journal

sequence. Together with the upper bound (i.e. maximum acceptance ratio) previously determined in Sec. III-B, the algorithm also maintains the lower bound which corresponds to the highest acceptance ratio so far and is adjusted whenever a better solution is found. Using the upper and lower bounds the algorithm then filters out sub-optimal embedding candidates, hence avoiding unnecessary computation if the solution quality cannot be improved. Moreover, the exploration can be stopped at anytime and the best solution thus far (the current lower bound) will be returned.

Without memoization, the algorithm would explore all $n!$ permutations of the VNR batch for a total of $n!n$ VNR processed. Considering the two bullet points above, memoization reduces the exploration to *all distinct permutations of all non-empty subsets* of the VNR batch. The number of such sequences can be calculated [3] as $\lfloor en! - 1 \rfloor$. This is the upper bound to the number of VNRs that will be processed, the actual number is lower due to the additional pruning described above.

E. VNE algorithm Pseudocode

The pseudocode of the VNE algorithm is shown in Algorithm 1. In addition to the existing symbols, we used $\mathcal{N}_k(n)$ to denote the k-hop neighborhood of node n; $\mathcal{H}(n)$ to denote the shortest-hop path from node n to sink; $\mathcal{C}_i$ to denote the candidate source nodes for VNR R_i; $\mathcal{W}$ denotes the weight of a link or path, or the weight of an element in the Knapsack problem; $\mathcal{A}$ represents the most VNRs that can be accepted, considering the limitation of the sink. For the Knapsack problem, the elements are the VNRs ($R_i \in \mathcal{R}$), the weight of each element set to the VNR requested quota (R_i^γ) multiplied by the Intra-Flow Interference factor at the sink (IFI), which depends on the length of the path. The best solution is represented as the mapped nodes $X^* = \{x_{ij}\}$ and links $Y^* = \{y_{ij}\}$.

Algorithm 1 VNE for WSN

Input: $G = (N, E), S, D, \mathcal{R}$
Output: $X^* = x_{ij}, Y^* = y_{ik}, i < |\mathcal{R}|, j \in N, k \in E$

// *Node mapping*
$\forall R_i \in \mathcal{R}, \mathcal{C}_i = \{n \in \mathcal{N}_2(R_i^s) | D_{nR_i^s} \leq R_i^\delta$ **and** $\mathcal{H}(n) min\}$
$x_{ij} = 1$ s.t. $\arg\max_{i<\mathcal{R}, j \in \mathcal{C}_i} |\cup \{j\}|$
// *Determining bounds*
0-1-K $(R_i \in \mathcal{R}, \mathcal{W}_i \leftarrow IFI(\left|\mathcal{H}(j|x_{ij} = 1)\right|) \times R_i^\gamma)$
$\mathcal{A} \leftarrow$ length of knapsack solution
// *Combinatorial search*
for all $P \in$ perm$(\mathcal{R})$
  $X = \{x_{ij}\}, Y = \{y_{ij} = 0\}$
  for all $R_i \in P$
    // *Link mapping for* R_i
    while true
      $p \leftarrow$ Dijkstra(src $= j | x_{ij} = 1$, dst=sink, $\mathcal{W} = e_i^\psi$)
      if $\mathcal{W}(p) \geq$ penalty
        found = **false**; **break**
      // *Reliability and capacity constraints*
      if $(\prod (e_i^\mu)^{y_{ij}} < R_i^\pi) || (e_k^\varphi > e_k^Z) \forall e_i \in p, e_k \in IN(p)$
        penalize worst link; **continue**
      found = **true**; **break**
    if found: $y_{ij} = 1 \forall j \in p$;update $e_i^\varphi, e_i^\psi \forall e_i \in E$
    else : $x_{ij} = 0$ // R_i *is not mapped*
    // *Pruning*
    if $\sum_X = \mathcal{A}$: **break** // *Sink bottleneck*
    if $\sum_X + (|P| - i) \leq \sum_{X^*}$: **break**
  if $\sum_X > \sum_{X^*}$
  or ($\sum_X = \sum_{X^*}$ **and** $\sum_Y e_i^\psi < \sum_{Y^*} e_i^\psi$):
    $X^* \leftarrow X$ **and** $Y^* \leftarrow Y$ // *Update current best*

F. Solution quality vs. processing time

The exploration of all the permutations of the input VNRs leads to finding the best embedding sequence and it is also a means to investigate the trade-off between processing time and solution quality. As explained above, as the search space is analyzed the algorithm keeps track of the current best solution and of the points in the search space where that is improved. The quality of the solutions is quantified using the acceptance ratio (ratio of input VNRs that are successfully embedded) and the embedding cost (sum of the link weights for the substrate state after the embedding is completed, as per Eq. 3). A solution is improved if i) the acceptance ratio is increased or ii) the acceptance ratio is the same but the cost is lower.

The following types of solutions are defined:

- **Best solution**, *HBest*, The solution resulting in the highest acceptance with the lowest embedding cost. Finding this solution requires processing the entire search space.
- **Good solution** Any solution which has as high acceptance ratio as the *best* solution and requires at most 10% greater embedding cost.
- **Initial solution** The solution representing the lower bound acceptance ratio. Obtaining this solution requires embedding only a single VNR sequence. It may or may not qualify as a *good* solution (might have lower acceptance ratio than the best).
- **Early solution**, *HEarly*, The first *good* solution (same acceptance ratio as the best) found during the exploration of the permutations.

These quantified solutions are used as milestones in the search for the best solution and they can be provided to the user as solutions of a bounded lower quality that can be obtained in known, bounded, time.

IV. Experiments and Results

We start by evaluating the performance of the algorithm compared to the MILP baseline and analysing the limitations imposed by the structure of the WSN. We then explore the trade-off between the solution quality and the processing time to identify the *quick & good* heuristic.

A. Evaluation baseline

The existing proposed solutions for VNE in WSN or wireless mesh consider only a subset of the constraints covered by our algorithm (i.e. either QoS or QoI, not both). Extending these solutions to make them equivalent to ours is not trivial

[3] https://oeis.org/A007526, see Joseph K. Horn's formula

and would require significant changes that could alter their performance so that it is not comparable with the original. We decided to compare the performance of the algorithm with an exact solution obtained by the Mixed Integer Linear Program shown in equations 1-6. To this extent, we modelled the MILP using the Pyomo framework [4] and used the Cbc (Coin-or branch and cut) solver [5].

As indicated by Rost *et al* [12] the common objective in VNE is either to maximize acceptance ratio or to minimize the resources required when accepting a subset of the VNRs (of a predefined minimum size). In comparison, the goal used in this paper is a combination of the two: maximizing the acceptance ratio while minimizing the cost (or the amount of substrate resources used). In accordance with the state of the art, the MILP in equations 1-6 was split into two separate problems. The first one, *MILP-acc*, maximizes the acceptance ratio given the input VNRs. The second one, *MILP-cost*, minimizes the cost. Both problems use the same set of constraints (equations 3-6).

The algorithm is compared to the baseline as follows. First, the *MILP-acc* is solved with Cbc to identify the maximum acceptance ratio for the input VNRs. To ensure fair comparison, only those configurations of input VNRs where the algorithm obtains the same acceptance ratio as *MILP-acc* (optimal acceptance ratio) are considered. For those input VNRs, the *MILP-cost* is solved, restricting the input VNRs to all the distinct subsets that are of the same size as the maximum acceptance value obtained by *MILP-acc* for that particular input.

B. Experimental setup

Our VNE algorithm takes as input a substrate topology $G(N, E)$ with the associated node and link properties, the measurement error matrix D and a set of VNRs. The algorithm evaluation was designed considering the following: $i)$ the algorithm's computational complexity depends primarily on the network size (number of nodes) as well as the number of VNRs, $ii)$ the outcome of embedding a given set of VNRs depends on the network topology as well as the Sensing Request Points (SRPs) requested in the VNRs; namely, some topologies or sink positions, or source locations can lead to better or poorer quality result. The VNE algorithm was evaluated with the network size varied over the set $50, 100, 150$ nodes and with 1-8 VNRs [6]. For each configuration of network size and number of VNRs, one thousand iterations were tested. To eliminate the bias noted in the second item above we randomized parameters in each iteration. The network topology and error matrix were randomly generated as explained below. The VNR parameters were uniformly sampled: i) SRP from the grid of SRPs of the corresponding topology; ii) required accuracy $\in (0,$ mean + std dev of error matrix values$]$; iii) quota $\in [1, 20]$ [7]. The minimum reliability for a VNR was set to 50%. The coefficients of the link weight (Eq. 1) were set empirically to $c_\varphi = 40$ for the link load and $c_\mu = 2$ for the packet loss. The chosen coefficient values prioritize valid solutions with balanced link load over valid solutions with reliability higher than requested. The former will lead to higher acceptance ratio, increasing profits; the latter will not.

The network topologies generated were designed to match organized WSN deployments, where groups of nodes are monitoring various rooms in a building or city blocks, and the groups are inter-connected to form a network without partitions. The reliability (packet success rate) of the link in the generated topologies was calculated based on the distance between nodes. As indicated in Section II, in a real setup the network topology and link qualities are obtained periodically from the network and are available to the algorithm in the network information databases.

For simplicity, it was assumed that all the nodes in the network have temperature sensors only. To generate the measurement error matrix D for each generated network topology a spatial distribution of temperature was simulated by adding sources of heat and cooling to the deployment area. Then, for each SRP and each node, the experienced temperature was calculated as the cumulative effect of all the temperature sources. That was then used to calculate the error matrix. This model ignores the variation of temperature in time. For increased accuracy a model based on covariance such as [13] can be used instead, however the choice of model does not affect the performance of the algorithm.

The algorithm and baseline implementation, as well as the input data for the experiments are available in our repository [8].

C. Performance evaluation

The quality of the embeddings obtained using the proposed VNE algorithm, in terms of acceptance ratio, embedding cost and processing time, was evaluated and compared to the baseline described in section IV-A. The performance of the algorithm was also evaluated in terms of overall network utilization and sink resource utilization. The evaluation considers the three types of solutions generated by the algorithm: best (*HBest*), early (*HEarly*), and initial.

A quick glance over figures 1, 2, 3 shows that the proposed algorithm generates solutions (*HBest*) that are similar in acceptance ratio and cost to the optimum (obtained by *MILP-acc* and *MILP-cost*). The heuristic is considerably faster than the MILP, with the early solution (*HEarly*) obtained in seconds.

Figure 1 shows the acceptance ratio obtained with the heuristic, compared to that obtained with the *MILP-acc* solver. The *HBest* acceptance ratio is represented with the boxes, three for each number of VNRs, corresponding to the results for 50, 100, and 150 node networks. The distance from the optimum (represented with lines and markers) is calculated as

[4]http://www.pyomo.org

[5]https://projects.coin-or.org/Cbc

[6]For higher values the processing time of the baseline became prohibitive.

[7]A 20% quota is the maximum that can be allocated due to a maximum IFI multiplier of five [7]. Considering an empirically determined link data rate of $624 Bytes/s$, the quota interval corresponds to application data rate $\in [6.24, 124.8] Bytes/s$. This can be determined based on application payload size and sampling rate as explained in section II.

[8]https://github.com/RKatonaAtNimbus/VNE_for_WSN

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3009834, IEEE Internet of Things Journal

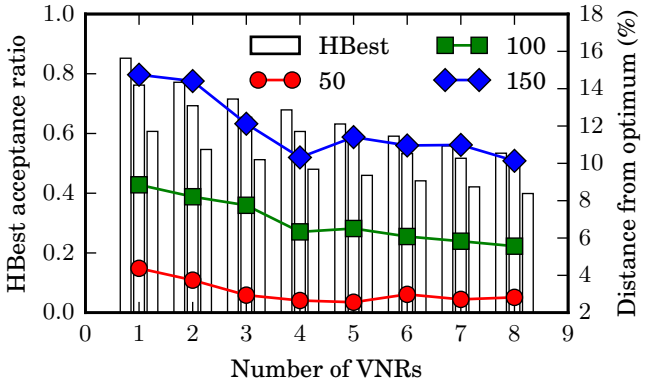

Fig. 1. Algorithm acceptance ratio (boxes, three for each number of VNRs, corresponding to 50, 100, and 150 nodes) and distance from optimum (lines).

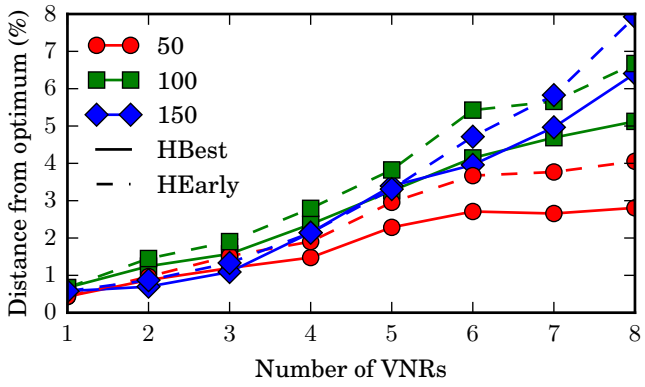

Fig. 2. Algorithm embedding cost relative to optimum.

$(milp - heur)/milp \times 100$ (*heur* is the heuristic solution). The plot shows that the acceptance ratio obtained by the algorithm decreases with the number of VNRs to less than 60% at 8 VNRs [9]. The same behavior is obtained by *MILP-acc*, although it is not shown in the figure. The primary cause is the use of a single sink, and it is analyzed in detail later in this section. The acceptance ratio also drops with increasing network size. This is due to larger networks having longer (and less reliable) paths, as well as higher probability of bottlenecks within the network.

The distance from optimum shown with lines and markers in Figure 1 increases with the size of the network due to the larger search space and greedy path finding algorithm. The values at eight VNRs are: 2.8% in 50 nodes, 5.5% in 100 nodes and 10.1% in 150 nodes. The figure also shows that the optimality gap decreases with the number of VNRs, however this is believed to be due to the averaging of the acceptance ratios: if *HBest* accepts one less VNR than *MILP-acc* the result can be an acceptance ratio of 0 for one VNR, or 0.875 for eight VNRs. To verify, Figure 4 shows the percentage of cases where *HBest* obtains optimal acceptance ratio. Intuitively, this value is proportional to the optimality gap. The trend observed here is indeed different (actually inverted) from what is shown in Figure 1, with the values decreasing (so optimality gap increasing) as the input size increases. The interpretation of the two figures 1 and 4 is that (at worst, for 150 node networks) the heuristic achieves optimal acceptance ratio in 60-65% of cases (Fig. 4). Where the result is suboptimal the optimality gap is of less than 15% (Fig. 1). The causes for the optimality gap are discussed in more detail in Section V.

In Figure 2, the embedding costs obtained by *HBest* as well as *HEarly* are compared with the optimum as obtained with *MILP-cost*. For fair comparison, the results are only from the cases where the acceptance ratio of the heuristic is optimal. As explained above, for each case, given the number of accepted VNRs (obtained by *MILP-acc*), *MILP-cost* was run over all the combinations of subsets of the entire input of length equal to the number of accepted VNRs to obtain the optimal (lowest) cost. The cost obtained by *HBest* is at most 6.4% greater than the optimum (for 150 nodes and eight VNRs). *HEarly*, which is defined to have the same acceptance ratio as *HBest* and at most 10% higher cost is at most 7.9% greater than the optimum.

[9]The reasons for obtaining less <100% acceptance for one VNR are rejections due to requested measurement accuracy not finding a valid candidate node, or a combination of measurement accuracy and reliability.

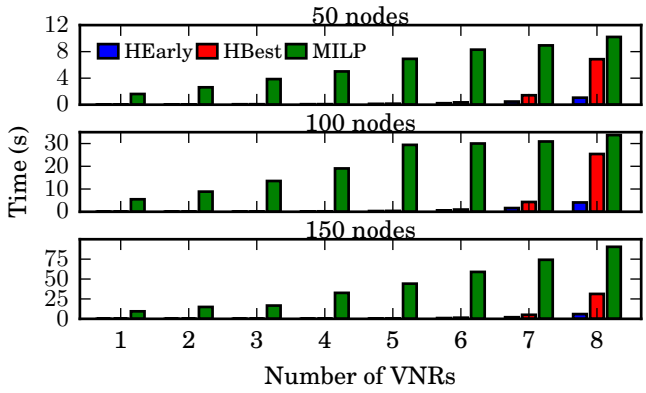

Fig. 3. Processing time (note the different y-axis scales).

The processing time of the heuristic *HBest* and *HEarly* solutions is compared to that of *MILP-acc* in Figure 3. The MILP representation of the VNE is known to be NP-Hard. The *HBest* solution has factorial time complexity because it

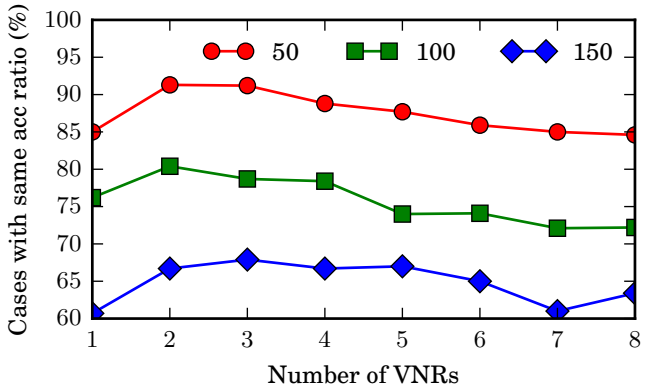

Fig. 4. Percentage of cases where the heuristic achieves optimal acceptance ratio.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3009834, IEEE Internet of Things Journal

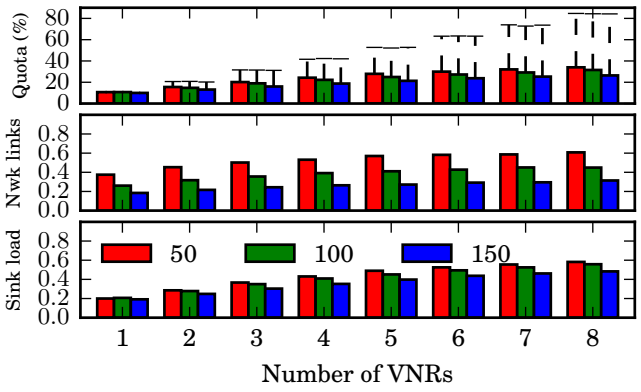

Fig. 5. Utilization of resources with regards to average quota.

embeds all the permutations of the input VNRs. Embedding a single sequence of VNRs has a worst case complexity of $\mathcal{O}(|\mathcal{R}|(|E|(|N|log|N|+|E|)))$, where N are the nodes, E the links, and $\mathcal{R}$ the VNRs, as per Section II [10]. Even with the gains obtained by memoization the *HBest* quickly becomes intractable. Nevertheless, the *HEarly* solution can generate results of the same acceptance ratio as *HBest* and within 10% of the *HBest* cost, but in much less time. All solutions increase in processing time with the size of the network and the number of VNRs, as expected. Although *HBest* and *MILP-acc* are close in processing time for the 50 and 100 node networks, at 150 nodes the MILP takes more than double the time. While the *HBest* could still be a viable alternative even in the 150 nodes at eight VNRs, the *HEarly* solution takes only a few seconds (<6s). Still faster is the *initial* solution that only embeds one sequence of VNRs, with a worst case of 0.2s (not shown in the figure). The quality of the initial solution is analyzed in the next subsection.

Improving the utilization of network resources is one of the main incentives of network virtualization, and the same goal is adopted for shared WSNs [6]. Figure 5 presents the network utilization. Starting in the top plot with the average quota requested by the VNRs (whisker lines) and average quota accepted into the network (bars), the results show that less than half of the requested quota is actually accepted, even though the amount requested is less than the full capacity of a link (<100% of the link's capacity). The figure presents communication quota rather than acceptance ratio because it is a more accurate measure of network utilization, however the results support those in Figure 1, where the acceptance ratio also decreases below 60%.

The middle plot of Figure 5 shows the percentage of network links that have been mapped to VNRs or affected by VNR traffic. Two trends are clearly identifiable. First, the percentage of mapped links decreases significantly as the network size is increased: at 50 nodes, 60%, at 100, 45%, and at 150, 31%. This is primarily because larger networks have more links, so the utilization will be relatively lower. Also, larger networks will have on average longer paths; these have lower end-to-end reliability (which decreases exponentially with path length), therefore lower likelihood of VNR acceptance.

The second insight is that the utilization, while increasing with the number of VNRs (and, consequently, with the demanded quota) has an asymptotic trend. The same trend can be observed in the load at the sink, in the bottom plot [11], and it is caused, as discussed in Section III-B, by the extraction of all the data through the sink. The sink becomes a bottleneck, limiting accepted quota and admission ratio. This is an important result that shows that using a single sink for a shared WSN is not effective and will not be able to maximize the utilization of the network. Multiple sinks should be used instead and they must be deployed strategically in the substrate; we are currently developing a solution to this.

D. Embedding quality vs processing time

As proposed previously, a sub-optimal but quickly obtainable *good* solution may be used instead of the *best* in order to reduce the required computation time. The advantage of the anytime algorithm defined in this paper is being able to explore the evolution of the solution quality with the processing time. Section III-F defined *best* (*HBest*), *early* (*HEarly*), and *initial* solutions, and the quality and processing time of the best and early solutions were analyzed in the previous subsection. The initial solution is the most attractive because it only embeds a single VNR sequence, and the longest processing time experienced in the experiments conducted was 0.2s. It is therefore important to quantify the quality of the initial solution, and this is analyzed in Figure 6 (worst case results obtained are shown). The stacked bars show a breakdown of the quality of the embedding results of the *initial* solution. It can be seen that in >90% of the cases, the initial solution qualifies as a *good* solution (which makes it equivalent to the *early* solution). Furthermore, in >60% of cases the *initial* solution is the *best* one. The important conclusion is that in 90% of the cases the algorithm can provide a solution with optimal acceptance ratio and within 10% of the optimal cost by only processing the input VNRs *once*, sorted in non-decreasing order of their requested communication quota. This result stands for batch sizes ≤ 8 VNRs, as tested, however the shape of the "Equals good solution" curve indicates asymptotic behaviour, which is a promising result for larger batch sizes.

For the cases when a *good* solution cannot be obtained immediately we analyze how much of the search-space needs to be processed in order to find one (*i.e. HEarly*). Figure 7 shows the distribution of the occurrence in the search space of the *HEarly* solution. The whiskers represent the 95th percentile and the y-axis is in log-scale. The results show that, for up to 8 VNRs, the *HEarly* solution will be found, with 95% certainty, if only the first several hundred permutations are explored.

[10]Embedding consists of running Dijkstra lowest cost path ($\mathcal{O}(|N|log|N|+|E|)$) and repeating after penalizing links (must try all links in the worst case), until success or all links are penalized. This must be repeated for all the VNRs in the sequence.

[11]A sink's load can reach 100% if its direct neighbors send continuously. In a multi-hop network, those neighbors need to release the channel to receive data from upstream, which reduces the maximum sink load to <100%.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3009834, IEEE Internet of Things Journal

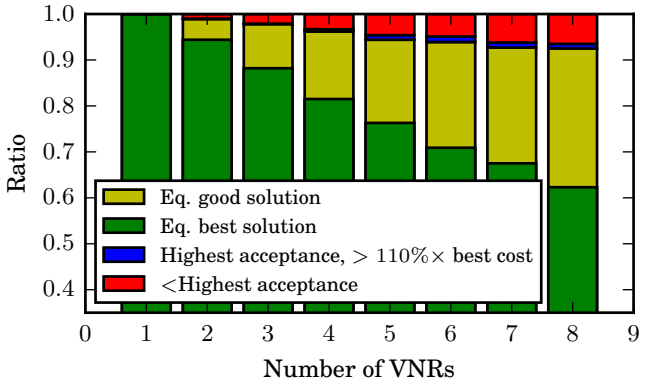

Fig. 6. The quality of the initial solution for 100 nodes substrate.

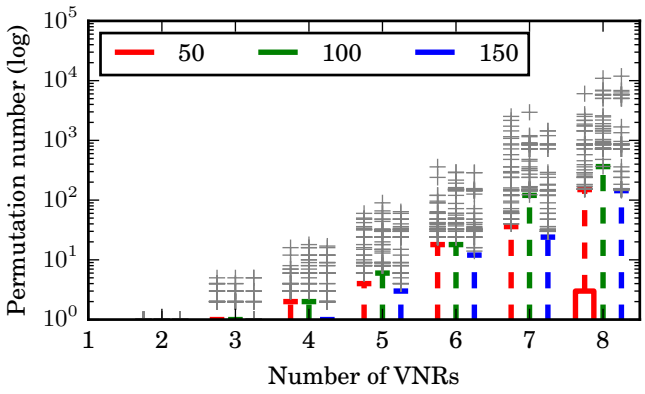

Fig. 7. Effort required to find the HEarly solution.

The behavior observed in the results above is explained by the objective function used. This maximizes VNR acceptance (primary) and minimizes embedding cost (secondary), so it favors requests that demand fewer communication resources (*i.e.* quota) and are closer to the sink. The initial solution will often be the *best* because it embeds the input VNRs in non-decreasing order of their quota. However, when the requested quotas are similar the cost or acceptance ratio can be improved in subsequent permutations of the input, due to the resulting total resource consumption or to violations of the reliability constraint.

V. Discussion

A. Assumptions and their limitations

Some of the assumptions that were made in the design of the experiments affect the results. Here we discuss the potential limitations imposed by our assumptions.

The networks generated have long paths. Long paths have lower reliability, so this can be seen as reducing the acceptance ratio. As indicated in Section IV-B the generated networks mimic large-scale clustered WSNs. The existence of long paths is not due to the network topology but primarily to the use of a single sink. Nevertheless, by randomizing the position of the SRPs we get both short and long paths, and capture a more general behaviour of VNE in WSNs.

One VNR is equivalent to one source node. While traditional network virtualization deals with embedding network topologies (e.g. in data centers), WSN applications are commonly source-sink, with potentially multiple sources. Therefore, we believe the assumption makes sense; to simulate multiple source networks we use multiple VNRs.

Eliminating bias. We carefully designed the experiments and the parameter value range to cover a wide range of WSN VNE scenarios and eliminate bias. The network topology is probably the most important, and by randomizing the topology, the location of the sink, and the location of the SRPs we eliminate any bias related to path length, in-network bottlenecks, or clustering of requests. We also covered a wide range of VNR requested accuracy and sampling rate values. By randomizing over 1000 cases, we obtained a result that represents the VNE performance over both common as well as more demanding (e.g. in terms of sampling rate) or relaxed (e.g. in terms of reliability) scenarios, which are uncommon but may still occur.

Scale. The goal of this work was to examine the trade-off between solution quality and processing time, due to the NP-Hardness of the problem. This resulted in the important result that with the initial solution we can obtain >90% best result in >90% of cases, for batch sizes of 8 VNRs or less. However, this required exhaustively exploring the search space, which imposed limits on the scale, in terms of network size and number of VNRs. Nevertheless, comparing with the state of the art, Yun *et al.* [14] consider networks of up to 50 nodes and the embedding is online, which means that VNRs are embedded one at a time (compared to batch embedding done in our algorithm). Delgado *et al.* [15] consider a network of 36 or 72 nodes, with up to 32 VNRs embedded. Finally, Afifi *et al.* [16] consider small networks of 18 nodes. While we consider a lower number of VNRs, we have a much higher (>2x) network size than the state of the art. This increases the search space, and considering that we also have more constraints, the resulting search time is considerably higher.

B. Distance from optimum

The algorithm presented in this paper obtains embeddings of within 10-15% of the optimal acceptance ratio and 8% of the optimal cost, as presented in the previous section. The optimality gap is caused by the node and link mapping procedures. Although the algorithm goes through all the feasible permutations of the input VNRs, the node and link mapping are greedy. Dealing with the suboptimality at these steps opens up interesting avenues for further exploration and improvement of the algorithm.

The node mapping is disconnected from the link mapping and always considers the entire set of input VNRs, regardless of the possible acceptance ratio. Since nodes are mapped trying to minimize the overlap (two virtual nodes on the same physical node), it is possible that some valid candidate nodes are avoided due to possible contention between two VNRs, even when one of those VNRs will not be accepted. Therefore, performing the node mapping for each VNR in tandem with the link mapping could provide better results.

Primarily the optimality gap comes from the path selection during link mapping. This can be seen in the single VNR

cases where the heuristic underperforms in acceptance ratio, so there is a viable path but the algorithm does not find it. It must be said here that the constrained shortest path problem is NP-Complete [11], and in our case it is further complicated by the link capacity constraint, which must take into account neighboring links. The approach used in this paper applies Dijsktra and then validates the path against the capacity and reliability constraints, penalizing the worst performing links. It is the penalization which can lead to sub-optimal solutions because a penalized link can actually belong to another valid path. We are currently exploring an extension of Dijsktra that checks the capacity and reliability constraints as the path is being built.

C. The VNE algorithm with a dynamic substrate

With few exceptions (*e.g.* [17]), most VNE algorithms assume a static substrate. In the real world this is rarely the case, especially in WSN. The VNE algorithm needs to provide a solution and enforce it into the network (configure nodes, deploy code, etc) before the substrate changes. The algorithm presented is an anytime algorithm so can handle this problem by returning the current best solution when preempted. The algorithm can continue searching for the optimal solution in the background and reconfigure the network (if needed) on completion.

D. Considering energy

Energy consumption is the primary constraint for WSN applications and the common solution is reducing the duty cycle. By definition the goal of VNE is somewhat opposed to that as the desire is to *maximize* the utility of the network. In allocating VNRs over the energy constrained substrate it is important to evenly distribute the load between nodes and prevent energy bottlenecks from forming that would deplete their energy faster and potentially partition the network. The algorithm presented distributes load both at the node mapping (minimizes overlap) and link mapping (minimizing link weight minimizes load as well) stages. Nevertheless, in this work controlling the energy consumption and network lifetime are only *side-effects* of the presented algorithms; in the future we plan to introduce energy consumption as an explicit constraint.

E. Impact of interference on network capacity

A valid embedding must ensure conflict-free transmission. This requires silencing all links within an active link's interference neighborhood [14]. For the VNE those link resources are consumed and cannot be allocated to requests. This results in network capacity degradation and ultimately limits how well the WSN substrate can be utilized.

The most visible impact of interference on link capacity is at the sink. As seen in Figure 5, the sink acts as a bottleneck, reducing the maximum accepted quota to $<100\%$ of a link's capacity. Nevertheless there may be other regions in the network suffering from congestion. For example, on longer paths, multi-access link interference can cause certain links on the path to be affected (*i.e.* their capacity *consumed*) up to 5 times the quota that is required for a single transmission, hence causing a bottleneck effect. In future work, we plan to experiment with varying network densities and multiple sink nodes to assess their effect on the embedding result.

VI. Related Work

The virtualization of WSNs has been discussed as a means to better utilize WSN deployments [18], [19], [20]. In addition to existing VNE constraints, VNE in WSN must ensure *reliable* and *correct* data extraction. Node allocation must be constrained by physical location and measurement errors (QoI), and link allocation by end-to-end communication reliability (QoS). Delgado *et al* [15], [21] and Bousnina *et al* [22] first explored heuristic and exact approaches to VNE in WSN with QoI, but without QoS. Bousnina *et al* [22] uses the same greedy link mapping as our algorithm, however, in their case, they only consider the least cost path. Our algorithm will explore higher cost paths, in case the least cost is not feasible, therefore it achieves higher acceptance ratio. Li *et al* [23] address VNE in industrial WSN with QoS, but without radio interference. Another resource allocation problem that is somewhat similar to VNE is that of mapping applications to nodes to maximize the quality of sensing [24]; this does not consider communication.

Research in VNE for wired networks has produced far more solutions [4]. According to [12] VNE algorithms either maximize the acceptance ratio or maximize the profit, given a set of acceptable requests [25]. Exact solutions defined as Mixed Integer Linear Programs (MILP) have been proposed [2] and shown to handle substrates of up to 50 nodes with VNs of up to 8 nodes in acceptable time. The problem can be simplified by reducing the MILP (known to be NP-Hard) to a Linear Program (LP) using randomized rounding. This has produced good solutions in polynomial time in [25], [26]. Finally, the problem can be simplified using heuristics [3], or by restricting the search space to subsets of the parameter space [27]. The existing breadth of VNE solutions for wired networks cannot be translated into the wireless domain, due to the way wireless interference impacts the constraints.

There are several ways for handling wireless interference and ensuring conflict-free communication. We took a similar approach to that of [15], ensuring that conflicting links are not active at the same time. This is a *sufficient* condition for conflict-free scheduling. Yun *et al* [14] show that a tighter bound can be obtained if the traffic patterns, that is the packet arrival times, are considered [14], [16]. However, this cannot be represented as a linear constraint but requires iterative analysis, which greatly increases the problem complexity. Cross-link interference can be evaluated using a protocol model [8], based on binary connectivity between nodes [14], or a physical model [8], based on SINR (Signal to Interference plus Noise Ratio) [28]. Another concern in wireless networks is the dynamic substrate that can change either due to node mobility or link variation. This was discussed by Abdelwahab *et al* [17].

VII. Conclusions

Based on our review of the state of the art, the algorithm we presented is the first Virtual Network Embedding algorithm for multihop wireless sensor networks that guarantees collision-free communication under QoS (end-to-end reliability) and QoI (measurement error) constraints. The algorithm provides solutions that are within 10-15% of the optimal acceptance ratio and within 8% of the optimal embedding cost.

The algorithm can be interrupted at any time and will return the best solution up to that point. While the time required to obtain the best solution is comparable to that for obtaining an exact solution with a Mixed Integer Programming solver, solutions of lower but still quantifiable quality can be found in bounded time. By exploring the improvement of the solution quality with the processing time it was found that in 90% of the tested cases a solution of equal acceptance ratio to the best, and within 10% of the best cost, can be obtained in bounded time (worst case 0.2s) by embedding the input VNRs sorted in non-decreasing order of their requested resources. The results show asymptotic behaviour indicating that the conclusion is likely to stand for larger batches as well.

The impact of the algorithm on the utilization of network resources was investigated. Probably the most important conclusion of the paper is that the improvement in resource utilization that network virtualization can provide for a single sink WSN and data collection queries is greatly limited. First, because all data needs to be extracted through the sink (no aggregation or in-network processing), which acts as a bottleneck, and second because ensuring conflict free communication on a link requires silencing neighboring (interfering) links. The possible solutions are using multiple sinks, data aggregation and in-network processing, and they will be explored in the future.

Acknowledgements. We thank Dr. Ramona Marfievici and the anonymous reviewers for their valuable comments.

References

[1] N. M. K. Chowdhury and R. Boutaba, "A Survey of Network Virtualization," *Comput. Netw*, vol. 54, no. 5, 2010.

[2] I. Houidi, W. Louati, W. B. Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *Comput. Netw*, vol. 55, no. 4, 2011.

[3] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, 2008.

[4] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual Network Embedding: A Survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, 2013.

[5] P. Corbalán, R. Marfievici, V. Cionca, D. O'Shea, and D. Pesch, "Into the smog: The stepping stone to centralized wsn control," in *Proc. MASS*, Oct 2016.

[6] D. O'Shea, V. Cionca, and D. Pesch, "The presidium of wireless sensor networks-a software defined wireless sensor network architecture," in *Proc. MONAMI*, 2015.

[7] J. Li, C. Blake, D. S. De Couto, H. I. Lee, and R. Morris, "Capacity of ad hoc wireless networks," in *Proc. MobiCom*, 2001.

[8] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, 2000.

[9] X.-Y. Li, P.-J. Wan, and O. Frieder, "Coverage in wireless ad hoc sensor networks," *IEEE Trans. Comput.*, vol. 52, no. 6.

[10] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. New York, NY, USA: John Wiley & Sons, Inc., 1990.

[11] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 7, 1996.

[12] M. Rost, S. Schmid, and A. Feldmann, "It's about time: On optimal virtual network embeddings under temporal flexibilities," in *Proc. IPDPS*, 2014.

[13] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *J. Mach. Learn. Res.*, vol. 9, no. Feb, 2008.

[14] D. Yun, J. Ok, B. Shin, S. Park, and Y. Yi, "Embedding of Virtual Network Requests over Static Wireless Multihop Networks," *Comput. Netw.*, vol. 57, no. 5, 2013.

[15] C. Delgado, J. R. Gállego, M. Canales, J. Ortin, S. Bousnina, and M. Cesana, "An Optimization Framework for Resource Allocation in Virtual Sensor Networks," in *GLOBECOM*, 2015.

[16] H. Afifi, S. Auroux, and H. Karl, "Marvelo: Wireless virtual network embedding for overlay graphs with loops," in *Proc. WCNC*, 2018.

[17] S. Abdelwahab, B. Hamdaoui, M. Guizani, and T. Znati, "Efficient virtual network embedding with backtrack avoidance for dynamic wireless networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 4, 2016.

[18] A. P. Jayasumana, Q. Han, and T. H. Illangasekare, "Virtual sensor networks-a resource efficient approach for concurrent applications," in *Proc. ITNG*, 2007.

[19] I. Leontiadis, C. Efstratiou, C. Mascolo, and J. Crowcroft, "Senshare: transforming sensor networks into multi-application sensing infrastructures," in *EWSN*, 2012.

[20] L. Sarakis, T. Zahariadis, H.-C. Leligou, and M. Dohler, "A framework for service provisioning in virtual sensor networks," *EURASIP J. Wirel. Comm.*, 2012.

[21] C. Delgado, M. Canales, J. Ortín, J. R. Gállego, A. Redondi, S. Bousnina, and M. Cesana, "Energy-aware dynamic resource allocation in virtual sensor networks," in *Proc. CCNC*, 2017.

[22] S. Bousnina, M. Cesana, J. Ortín, C. Delgado, J. R. Gállego, and M. Canales, "A greedy approach for resource allocation in virtual sensor networks," in *Proc. WD*, 2017.

[23] M. Li, C. Hua, C. Chen, and X. Guan, "Application-driven virtual network embedding for industrial wireless sensor networks," in *Proc. ICC*, 2017.

[24] S. Bhattacharya, A. Saifullah, C. Lu, and G.-C. Roman, "Multi-application deployment in shared sensor networks based on quality of monitoring," in *Proc. RTAS*, 2010.

[25] M. Chowdhury, M. R. Rahman, and R. Boutaba, "Vineyard: Virtual Network Embedding Algorithms with Coordinated Node and Link Mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, 2012.

[26] M. Rost and S. Schmid, "Charting the complexity landscape of virtual network embeddings," in *Proc. IFIP NETWORKING*, 2018.

[27] J. Lu and J. Turner, "Efficient Mapping of Virtual Networks Onto a Shared Substrate," WUSTL, Tech. Rep. WUCSE-2006-35, 01 2006.

[28] G. Di Stasi, S. Avallone, and R. Canonico, "Virtual network embedding in wireless mesh networks through reconfiguration of channels," in *Proc. WMCNC*, 2013.