Approximated Pareto Analysis for Fast Optimization of Large IEEE 802.15.4 TSCH Networks

Hamideh Hajizadeh, Rasool Tavakoli, Majid Nabi, and Kees Goossens
Department of Electrical Engineering, Eindhoven University of Technology, the Netherlands
{h.hajizadeh, r.tavakoli, m.nabi, k.g.w.goossens}@tue.nl

***Abstract*—The IEEE 802.15.4 Time-Slotted Channel Hopping (TSCH) is a widely used standard technology for industrial Wireless Sensor Networks (WSNs). The applications of such networks have diverse Quality-of-Service (QoS) demands that should be satisfied. Optimal configuration of the network parameters based on their QoS requirements and run-time adaptation to continuously meet the QoS requirements given all network dynamics is a great challenge for large-scale networks. The configuration space is very large for large-scale networks resulting in space exploration to be complex and extremely time-consuming. Moreover, such space exploration needs to be performed multiple times at design-time to give insight into the worst and best case performance of the mechanisms and network topology. Yet, it is needed for the run-time reconfiguration upon changes in the network. To address the stated challenges, we propose a fast and accurate enough algorithm based on Pareto algebra to extract a subset of Pareto configurations for large TSCH networks in a very short time. A proper configuration can be then picked from this set. Having such a fast optimization algorithm, the network can react to the changes in the link quality, routing topology, and reconfigure itself optimally at an appropriate time. The performance of the proposed technique is extensively evaluated, and compared with other approaches such as basic incremental Pareto analysis, and genetic algorithm, showing its superior execution time and quality of configurations in terms of accuracy and diversity.**

***Index Terms*—IEEE 802.15.4 TSCH, WSNs, Pareto analysis, Multi-objective optimization**

I. Introduction

Wireless Sensor Networks (WSNs), consisting of sensor devices connected through wireless links, are widely employed in many application domains. The IEEE 802.15.4 [1] protocol standard specifies the Physical (PHY) and Medium Access Control (MAC) layers of low-rate, and low-power WSNs. Time-Slotted Channel Hopping (TSCH) as a MAC operational mode of this standard is proposed to overcome persistent interference and multi-path fading using channel hopping technique. Moreover, TSCH provides predictable and reliable communications and increased network capacity due to time-slotted and multi-channel communications. Accordingly, these networks have received strong attention in industrial applications running in harsh environments [14]. The TSCH protocol supports two medium access methods using dedicated timeslots assigned to a link for transmission in a collision-free manner, and shared timeslots for communications of multiple links. The CSMA/CA mechanism is used to handle contentions and repeated collisions in the shared timeslots.

Recent studies such as [7], and [3] have shown that the selection of CSMA/CA MAC parameters of a TSCH network significantly influence the network performance. Therefore, it is very important to properly configure these parameters to get the best performance of the network operation both at design-time as well as at run-time. However, in the most real-world applications today, the parameter settings are mostly selected based on experience and rules of thumb, which results in non-optimal performance far off the QoS demands. Even if the network is configured optimally at design-time, it does not guarantee to be optimal under various channel conditions or changes in the routing topology, and scheduling mechanisms, specifically in harsh environments with dynamics. To tackle this problem, several works e.g., [16] and [11] focus on run-time adaptation and self-configuration of the parameters in these networks to deal with dynamics or changes in the network. They have introduced platforms including optimization engine (or optimization solver) as a main core of the platform to find optimal configurations using the state-of-art algorithms.

The main problem that is not addressed in these works is the speed of the optimization engine to produce the optimal configurations and its performance in terms of accuracy and diversity of the derived configurations. For example in [11], genetic algorithm is employed as multi objective optimization method which is not fast enough for run-time execution especially when the network scales up. This problem may cause the system to fail in setting suitable configurations in real-time, and consequently fail to satisfy the application requirements. Moreover, at design-time, the design space of the whole network for various available mechanisms, routing topologies, as well channel conditions needs to be explored in order to get sufficient insight into the network performance at run-time. This requires a massive amount of computation, and is not feasible when the network gets a bit large. Therefore, a fast and still high-quality optimization technique is of paramount importance for design-time decisions as well as run-time adaptation and self-configuration of the parameters in a TSCH network.

To approach this problem, we present a scalable multi-objective optimization method to extract a set of MAC parameters that lead to (near-)optimal performance of a TSCH network. Here, we consider a cluster-tree TSCH network with a converge-cast application. Without loss of generality, Orchestra scheduling [5] is used to build the TSCH schedule for the network. Our proposed approximated method provides optimal configurations for MAC with less than 1% error compared to the true Pareto configurations that is generated using the algorithm in [10]. Moreover, for large networks, our method

completely dominates the results of the genetic algorithm in terms of speed, accuracy, and diversity of the configurations while the algorithm in [10] cannot even complete the process of generating of the Pareto configurations due to memory limits.

The rest of the paper is organized as follows. Related work about optimization of the parameters in TSCH networks is surveyed in the next section. An overview of the TSCH CSMA/CA mechanism and Pareto algebra is given in Section III. Section IV presents the problem statement. The approximated Pareto analysis to optimize the performance of TSCH networks is presented in Section V. Performance evaluation setups and results are discussed in Section VI. Section VII concludes.

II. Related Work

This section provides an overview of the most relevant efforts in the literature for optimization of TSCH-based networks.

Authors of [15] propose a method that finds the optimum scheduling based on the shared timeslots in order to meet the QoS constraints while minimizing the energy consumption. [2] provides a reinforcement learning technique to optimize the performance of channel hopping in TSCH networks. Multi-Armed Bandit (MAB) algorithm as a learning algorithm is used to select the high-quality channels through collected data via a platform. The results show that the performance of the TSCH network in terms of reliability and energy consumption is improved compared to the default TSCH operation. In [8], the authors investigate the impact of the slotframe length on the energy consumption in a TSCH network. They also present a statistical analysis to find the relation between the energy consumption, the slotframe length, and topology of the network. The optimal slotframe length for different topologies to minimize energy consumption is then derived.

In [4], the problem of optimal routing in TSCH networks is addressed. The authors state a multi-objective optimization problem as a function of the schedule length, the packet routing tree cost, and the hop count which affects the end-to-end latency of the network. [13] provides scheduling methods for throughput optimization of a TSCH network. Two approaches are investigated to find scheduling methods that optimize throughput in two cases. The first approach considers the scenario that the statistical information of the channel state is available a priori. The second approach assumes that no channel state knowledge is available at design-time. For this case, a machine learning algorithm is presented by modeling the scheduling problem in terms of a combinatorial multi-armed bandit (CMAB) process.

None of these works in the literature focus on optimizing the CSMA/CA parameters to derive an optimal performance of large-scale multi-hop TSCH networks. Moreover, the majority of these works focus on single-objective optimization problems to find the single globally optimal solution. Such single-objective optimization problems may not support QoS for real WSN applications, since it may extremely impair other quality metrics of the WSN. Thus, for realistic WSNs, it is reasonable to optimize the overall performance considering multiple objectives such as reliability, latency, and network life-time, which leads to a set of Pareto-optimal solutions instead of a single optimal solution. Having multiple objectives, the optimization problem gets computationally too expensive, demanding very fast optimization techniques.

Several works employ heuristic algorithms such as genetic algorithm [11] as multi-objective optimization algorithm to produce Pareto optimal configurations. However, execution time of genetic algorithms to derive good quality results is high for large-scale networks. Furthermore, it does not guarantee to generate full-set of Pareto optimal configurations due to its random behaviour.

Authors in [10], propose an accurate multi-objective optimization method for cluster-tree WSNs, which outputs full-set of Pareto optimal configurations. They show that the Pareto configurations of different clusters of a tree-based WSN can be incrementally combined to obtain the Pareto configurations of the whole network with optimal end-to-end performance. Inspired by this work, we first use incremental Pareto combination algorithm to optimally configure CSMA/CA parameters in a multi-hop TSCH network, which is not covered in the literature. However, this can still lead to high computation overhead. To overcome this, we then propose an approximated Pareto analysis that derives a subset of Pareto configurations of the whole network with satisfactory accuracy and diversity in a very short time.

III. Backgorund

A. TSCH CSMA/CA Mechanism

The TSCH protocol uses CSMA/CA back-off mechanism in shared timeslots to avoid repeated collisions. A node with a packet in the MAC transmission queue transmits it in the first available shared timeslot. If no acknowledge packet (ACK) is received, a back-off procedure is activated and the back-off parameters are initialized. This includes setting the Back-off Exponent (*BE*) to its minimum value of BE^{min}, and the number of retransmissions (*RT*) to zero. Then, the node waits for a random number of shared timeslots in the range of $[0, 2^{BE} - 1]$ and transmits its data packet. If ACK is not received, *RT* is increased by one to count the number of retransmissions, and *BE* is increased by one up to its maximum value of BE^{max}. The back-off mechanism continues until ACK is received or a maximum number of retransmissions specified by *maxR* is reached.

BE^{min}, BE^{max}, and *maxR* are the MAC parameters that should be configured in the network. Each configuration has a different effect on the performance of the network in terms of communication reliability, latency, and energy consumption. In this paper, we aim to propose a scalable algorithm to derive all the possible configurations for a TSCH network that lead to optimal performance of the network.

B. Pareto analysis overview

Pareto analysis is an approach for multi-objective optimization based on the performance trade-offs [6]. This method can be used for design space exploration to derive the optimal configuration sets, called Pareto points.

Assume a system with p configuration parameters, indicated by $P_1, P_2, ..., P_p$ and q quality metrics indicated by $Q_1, Q_2, ..., Q_q$. Quality metrics present the performance of the system that are affected by the configuration parameters.

Configuration Space: It is the set of all possible configurations of the system and is the cartesian product of finite configuration parameters, indicated by $C = P_1 \times P_2 \times ... \times P_p$. $\bar{c} = (c_1, c_2, ..., c_p)$ is an element of such a configuration space which leads to the performance $\bar{Q}(\bar{c}) = [Q_1(\bar{c}), Q_2(\bar{c}), ..., Q_q(\bar{c})]$.

Dominance: For any two different configurations $\bar{c_1}$ and $\bar{c_2}$, $\bar{c_1}$ dominates $\bar{c_2}$ *iff*:

$$\bar{c_1} \succeq \bar{c_2} \Leftrightarrow \forall\ 1 \leq k \leq q, \quad Q_k(\bar{c_1}) \succeq Q_k(\bar{c_2}) \qquad (1)$$

This means that the performance of $\bar{c_1}$ is better or equal to performance of $\bar{c_2}$ for all quality metrics. Thus, we can remove $\bar{c_2}$ from the configuration space for optimization purposes in order to make the search space smaller [10].

Pareto optimal: The set $C^{opt} \subseteq C$ of configurations is Pareto optimal (Pareto minimal) if none of the elements in C^{opt} is dominated by other members. In other words, for any two different configurations $\bar{c_1} \in C^{opt}$ and $\bar{c_2} \in C^{opt}$, $\bar{c_1} \not\succeq \bar{c_2}$. Moreover, for all elements in C, there is an element in C^{opt} that dominates it. C^{opt} includes all Pareto optimal configurations, called Pareto points. For optimization purposes, we can use the Pareto optimal set instead of the whole configuration space.

Monotonicity: A function $f : C \rightarrow Q$ is monotone if for any two different configurations: $\bar{c_1} \succeq \bar{c_2}$, $f(\bar{c_1}) \succeq f(\bar{c_2})$.

Pareto Combination: Assume a system denoted by S that includes two subsystems S_1 and S_2, and configuration spaces C_1 and C_2, respectively. These configuration spaces can be combined into one joint configuration space C of S using cartesian product operation ($C = C_1 \times C_2$). In the process of combining two sub-systems, if the function that maps the configuration space C to the quality metrics Q of S is monotone, then:

$$C_S^{opt} \subseteq C_1^{opt} \times C_2^{opt} \subseteq C_1 \times C_2 = C \qquad (2)$$

It means that the Pareto optimal set of C is the Pareto optimal set of the Cartesian product of Pareto optimal set of C_1 and C_2 [10].

IV. Problem Statement

A. System model

Assume a WSN with *N* wireless sensor nodes, in which each node intends to send its data packets to a sink node. Each node runs the TSCH standard as the MAC layer, and a cluster-tree mechanism such as RPL [12] as its multi-hop routing protocol. Without loss of generality, the Orchestra scheduler [5] in the receiver-based mode is used to schedule TSCH timeslots. One timeslot in each slotframe is shared between all the nodes in each cluster to transmit their data packets to their parent, called Cluster Head (CH). Accordingly, the length of the slotframe is at least equal to the number of clusters in the network. This is illustrated in Fig. 1 for a small multi-hop network, in which three shared timeslots in the slotframe are assigned to three clusters.

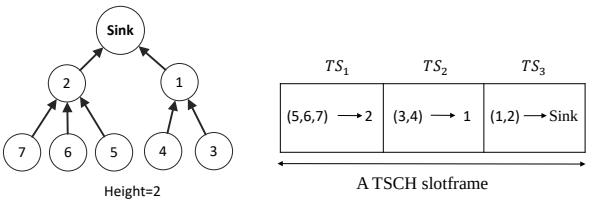

Fig. 1. An illustrative example of the assumed system model

To derive the performance of each cluster in terms of reliability, latency, and energy consumption per link, the proposed model in [7] is used. From the user's point of view, it is the end-to-end performance metrics that should be satisfied. Thus, we derive end-to-end performance metrics by combining the cluster level performance metrics.

In our system model, we assume a converge-cast WSN in which each node sends its data packet to its CH. Each CH aggregates all the received packets from its children periodically and sends it to its parent. This process continues until all data items reach the sink node. Thus, it is important to consider the end-to-end performance metrics as QoS constraints. Here, without loss of generality, we consider the worst-case end-to-end performance (we can also consider average-case end-to-end performance, or end-to-end performance per each node due to monotonicity being satisfied) in the network for optimization.

- ***PEP*** is the worst-case end-to-end packet error probability in the network. As multi-hop communications are used, *PEP* of any multi-hop route in the network can be calculated by considering the *PEP* of all the hops, as:

$$PEP = \max_{route \in L} \big(1 - \prod_{hop \in route} (1 - PEP_{hop})\big), \qquad (3)$$

  where L is the set of routes from all source nodes in the network to the sink node.
- ***PL*** denotes the worst-case end-to-end packet latency in the network. As we use aggregation at intermediate nodes on a multi-hop route, latencies of all the hops on a route can be added to extract the *PL* for that route. Hence, *PL* can be calculated as:

$$PL = \max_{route \in L} \sum_{hop \in route} PL_{hop}. \qquad (4)$$

- ***E*** indicates the worst-case energy consumption by the nodes in the network. This is the maximum energy that is consumed for a packet transmission by any node, on any route in the network.

$$E = \max_{route \in L} \big(\max_{hop \in route} E_{hop} \big) \qquad (5)$$

For each cluster in the network with a given configuration, we extract PEP_{hop}, PL_{hop}, and E_{hop} using the TSCH performance model in [7]. Here, *PEP*, *PL*, and *E* represent the worst cases of reliability, latency, and energy consumption for a specific configuration of the network, respectively. We consider these performance metrics as three objectives for the network optimization.

TABLE I
RESULTS OF IPCA FOR DIFFERENT NETWORKS

N	$\|IPCA\|$	$t_{IPCA}(s)$
9	1369	9.25
15	4802	86
27	4766	404
48	13893	14097
60	13795	15416
72	-	-

$|X|$: number of Pareto points using algorithm *X*.
t_X: the execution time of algorithm *X*.
$N_{CH} = N/3$, N_{CH} is the number of clusters.

B. Applicability of Incremental Pareto Combination Algorithm

In this paper, we aim to optimize the performance of a TSCH network by optimally configuring CSMA/CA parameters. We first start with the same approach that is described in [10], called Incremental Pareto Combination Algorithm (IPCA), since objective functions described in the system model are monotone. This algorithm starts by initializing all nodes as one-node clusters. Subsequently, clusters are combined incrementally step by step into larger clusters. In this algorithm, Pareto points (including quality metrics and configuration parameters) are produced at each step and forwarded to the next step. In the next step, Pareto optimal set of cartesian product of Pareto configurations from the previous step are considered as Pareto optimal set of the new cluster's configuration. This minimization process at each step reduces the number of new cluster's configurations, which results in much faster Pareto analysis. Following the same approach in our system model as a cluster-tree network, we tried to derive Pareto optimal configurations of TSCH networks. According to the standard, CSMA/CA parameters obtain values in the range of $0 \leq maxR \leq 7$, $3 \leq BE^{max} \leq 8$, and $0 \leq BE^{min} \leq BE^{max}$. This results in 312 configuration settings per cluster (assuming identical configurations for all the nodes in a cluster). We implemented IPCA in Matlab to produce Pareto points of the several tree networks including 9, 15, 27, 48, 60, and 72 nodes with degree of 3. Table I shows the number of Pareto points and execution time of IPCA for these networks. According to the results in Table I, the execution time of IPCA increases by expanding the network to more clusters. For network sizes bigger than 72, the IPCA methods failed to result in the final Pareto set for the whole network due to memory overflow and timeouts. This shows that the basic IPCA is still not scalable enough for large TSCH networks although it is a promising technique that is proven to be able to produce true Pareto configurations. This problem may give rise to a more serious issue when we intend to perform run-time optimization of the CSMA/CA parameters to cope with quick changes in the network and consistently satisfy the QoS requirements.

V. APPROXIMATED PARETO ALGORITHM (APA)

In this section, the approximated Pareto algorithm, as the main focus of this paper, is described as a solution to find Pareto-optimal configurations of large-scale TSCH networks in a short time to support run-time optimization platforms. The approximated method must consider two factors that are the key components of all multi-objective optimizations. Firstly, the approximated Pareto set must produce Pareto quality metrics as close as possible to those of the true Pareto optimal set. Secondly, it must provide Pareto quality metrics that are as diverse as possible.

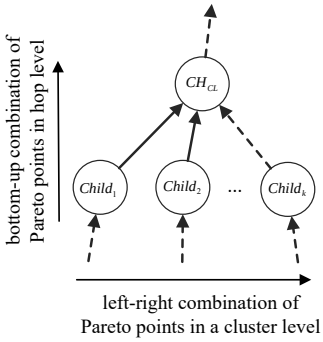

Fig. 2. Bottom-up and left-right combination of Pareto points

In IPCA, the number of extracted Pareto points at each step can extremely affect the complexity and computation time of IPCA. Therefore, it is desirable to smartly select a limited number of Pareto points at each step and forward them to the next steps to be combined. However, this selection should not cause loss of diversity in Pareto points in all dimensions as well as their qualities. To do so, we pick one Pareto point from those that possess very similar quality metrics. For example, suppose that $[0.2, 70, 0.9]$, $[0.199, 75, 0.901]$, and $[0.3, 55, 0.7]$ are Pareto points of the performance in a cluster ([*PEP*, *PL*, *E*]), in which *PL* and *E* are in timeslot and mJ, respectively. We can remove one of first two points and only forward the two remaining points to the next step to be combined with the upper cluster's Pareto points. More precisely, a 3-D vector including three parameters, $r = [r_1, r_2, r_3]$, is defined as a closeness factor in which r_1, r_2, and r_3 indicate closeness in *PEP*, *PL*, and *E* dimensions, respectively. For each Pareto point, we consider a 3-D closed volume around the Pareto point that bounded by r_1, r_2, and r_3, and keep only one Pareto point inside the volume. This selected Pareto point is the one that has the first order in the sequenced of configuration settings. Selection of r_1, r_2, and r_3 as closeness parameters makes a speed-quality trade-off. It is in the scope of the designer who decides how much the quality (accuracy and diversity) of the configurations is traded off for speed of the algorithm according to the given application.

Fig. 2 shows a typical cluster in a tree with cluster head CH_{CL} and children of 1 to k. Let, C_{CL} and C_{CL}^{opt} , ($1 \leq CL \leq N_{CH}$), be the configuration space and Pareto optimal configurations of cluster *CL*, respectively. Generally, for each child k in the cluster *CL*, C_k^{opt} achieved from the previous step is combined with C_{CL}^{opt}. For each combined configuration, reliabilities are multiplied, latencies are added and the maximum energy consumptions are considered (we call it bottom-up combination). Looking into the outcome of the bottom-up combination, only those are selected in which the children of cluster *CL* have the same configurations. This is a direct result of the assumption that all nodes in a cluster

have identical configurations. These selected configurations are then combined through Cartesian product. For this minimized configuration space of cluster *CL*, defined as C_{CL}, the maximum values of performance metrics are reported as the worst-case performance (called left-right combination). Pareto optimal set of C_{CL} is extracted through Pareto analysis, one of the close Pareto points is selected according to given r values and assigned to C_{CL}^{opt}. We continue this process of combining the clusters until reaching the sink node and obtain $C_{CL}^{opt}(sink)$, which includes the Pareto optimal configurations of the whole network. Incremental combination of Pareto points has also the ability to apply QoS constraints at each step prior to combining the Pareto points, which results in much faster analysis. This is in line with the fact that if reliability, latency, and energy consumption are not satisfied within a cluster of the network, the end-to-end metrics definitely cannot be satisfied in the whole network.

Algorithm 1 describes the steps of APA inside the Main function, which starts from line 9. The first loop traverses over clusters from 1 to N_{CH}, and the inner loop iterates over all possible configurations. The performance of the clusters for each configuration are derived using the performance model in [7] (line 13). If the performance provided by the cluster configuration does not meet the user's constraints, it is removed from the eligible configurations. Then, the algorithm calls the Get_Pareto function (line 17) that returns the Pareto points for each cluster. In the Get_Pareto function, first the non-dominated points are removed. Then, among all Pareto points that their performance metrics are within a 3-D volume with dimensions $r_1 \times r_2 \times r_3$ (called r-close points), the first one in the sequenced order of configuration settings is selected and the others are removed. In the second part of the algorithm, starting from line 18, incremental combining of the Pareto points begins. It is performed by combining the clusters from the bottom to the top levels of the tree structure until it reaches the sink. In each level and for each cluster, Pareto points are combined with its upper cluster's Pareto points (bottom-up combination) specified in lines 20 to 22. Then, Pareto points of the children in the clusters are combined by the left-right combination in IPCA. These two combinations result in considerable reduction of the size of the configuration space of the clusters in each level, C_{CL}. Again, the configurations that do not satisfy the QoS constraints are removed from the configuration space in line 23. Afterwards, Pareto configurations are derived, and r-close Pareto points are removed by running the Get_Pareto, and assigned to Pareto optimal configurations of the cluster. This procedure continues until it reaches the sink node, which provides the Pareto configurations of the whole network.

This algorithm may lose some Pareto points at each step and add some error to the results. However, due to the diverse selection of the Pareto subsets at each step, the approximated Pareto set represents the original Pareto set with a good accuracy. In the Algorithm, parameter selection for the elements of vector *r* in line 6 determines the quality of the selected Pareto points at each step of the algorithm. Larger values for the elements of *r* lead to fewer Pareto points, faster execution time, and lower accuracy.

Algorithm 1: The Approximated Pareto

```
1  Get_Pareto (C{}, Q̄{})
       /* Extracting Pareto points                          */
2      for each c̄_k ∈ C do
3          if ∃ c̄_j ∈ C | Q̄(c̄_j) ≻ Q̄(c̄_k) then
4              C(c̄_k) ← {};
       /* Removing close Pareto points                     */
5      for each c̄_k ∈ C do
6          ∀ c̄_j ∈ C | j ≠ k & ‖ Q̄(c̄_j) − Q̄(c̄_k) ‖≤‖ r ‖ do
7              C(c̄_j) ← {};
8      return C;
9  Main ()
10     for CL=1 to N_CH do
11         Q̄_CL = [ ] , C = [ ]
12         for each c̄_k ∈ C_CL (1 ≤ k ≤ 312) do
               /* using the model in [7]:                  */
13             Q̄(c̄_k) ← [PEP_CL, PL_CL, E_CL];
14             if Q̄(c̄_k) ⪰ QoS_req then
15                 Q̄_CL ← Q̄_CL ∪ {Q̄(c̄_k)};
16                 C ← C ∪ {c̄_k};
17         C_CL^opt ← Get_Pareto(C, Q̄_CL);
       /* Pareto combination                               */
18     for each level from down up to Sink do
19         for each cluster CL ∈ level do
               /* Bottom-up combination:                   */
20             for each child node k ∈ CL do
21                 C_k ← C_CL^opt × C_k^opt;
22             /* Right-left combination:                  */
                   C_CL ← ∏_{k∈CL} C_k
23             ∀c̄_k ∈ C_CL | Q̄_CL(c̄_k) ≺ QoS_req do
24                 C_CL(c̄_k) ← {};
25             C_CL^opt ← Get_Pareto(C_CL, Q̄_CL);
```

VI. Results

We implemented our proposed algorithm, APA, in Matlab to produce Pareto points of several tree networks running TSCH protocol. To evaluate the performance of APA as an optimization method, two factors are considered: closeness to the reference Pareto front, and spread along the Pareto front. Here, we use distance-based Performance Indices (PI) [9] to compare the accuracy and spread of the derived Pareto set. Given that P is the Pareto set produced by APA, and S is the reference Pareto set, Pareto Front Error (*PFE*) of any point $p \in P$ is defined as the distance of p from the closest point in S. It is used to present the accuracy and diversity of a Pareto set and is defined as

$$PFE(p) = \min_{s \in S}(d(p, s)) \tag{6}$$

where $d(p, s)$ is Euclidean distance between p and s. Distribution of *PFE* shows the spread of a given Pareto set along the reference Pareto set [9]. *MPFE* is the worst-case *PFE* calculated by

$$MPFE = \max_{p \in P} PFE(p). \tag{7}$$

This metric is introduced as an accuracy PI to compare the quality of the produced Pareto sets with the reference Parto set.

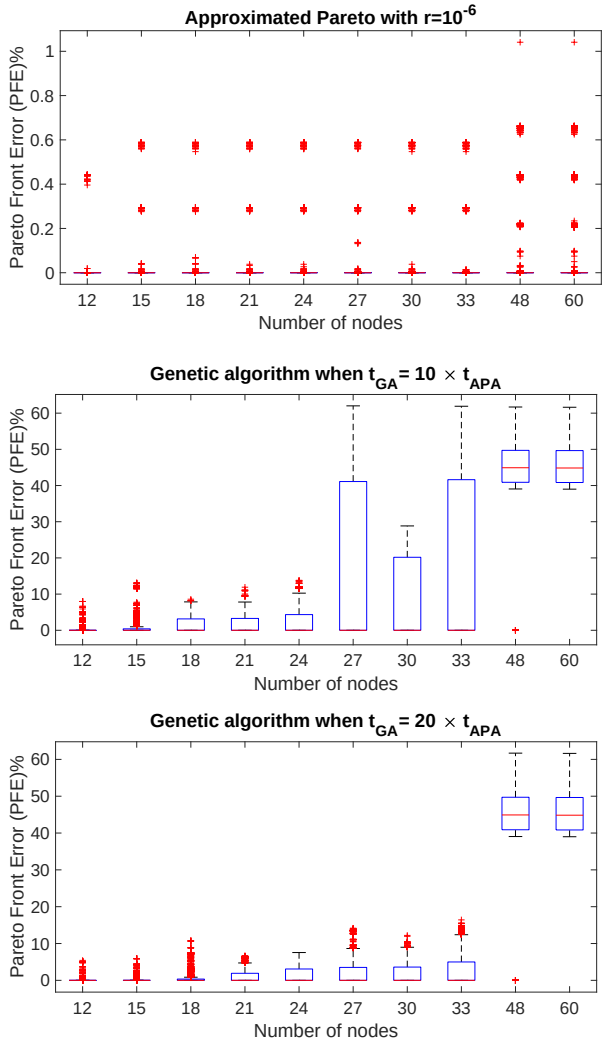

Fig. 3. Distribution of *PFE* for APA and GA

Here, for simplicity, *PEP*, *PL*, and *E* are normalized to their maximum values to be in the range of $[0, 1]$, and the closeness vector's elements are assumed to be the same. Hence, in the following r represents r_1, r_2, and r_3 ($r = r_1 = r_2 = r_3$). Moreover, the performance of APA in finding the Pareto optimal set for different scenarios is investigated. TSCH networks with tree structure are considered to minimise [*PEP*, *PL*, *E*], in which *PL*, and *E* are in timeslot and mJ, respectively. The performance constraints are set as 0.3, $10 \times$ *height* $\times N_{CH}$ timeslots, and 0.5 mJ, respectively (*height* is the height of tree for each network).

For several tree networks including 12, 15, 18, 21, 24, 27, 30, 33, 48, and 60 nodes with degree of 3, Pareto optimal set is derived through APA, while r is set as 10^{-6}. IPCA also runs in Matlab to produce reference Pareto set for comparison. Moreover, genetic algorithm (GA) is implemented to compare with APA in terms of accuracy and spread along the reference Pareto front, since genetic algorithm is mostly used as multi objective optimization algorithm for run-time optimization platforms such as [11]. Figure. 3 shows the distribution of *PFE* in percentage to illustrate the accuracy and spread of the

TABLE II
PERFORMANCE COMPARISON OF APA WITH $r = 10^{-6}$ AND GA COMPARED WITH IPCA

N	$t_{APA}(s)$	$MPFE_{APA}\%$	$MPFE_{GA_{10}}\%$	$MPFE_{GA_{20}}\%$
15	15	0.58	13.19	5.92
21	22	0.58	12	6.72
27	27	0.58	62	14.16
33	36	0.58	61.1	16.42
48	67	1	61.6	61.7
60	87	1.04	61.6	61.62

derived Pareto sets. (Since performance metrics are normalised to their maximum values, *PFE* is in the range $[0, \sqrt{3}]$). GA is run for 10 and 20 times (GA_{10} and GA_{20}) the APA's execution time to generate Pareto optimal set. Table II also reports the execution time of APA and MPFE of three cases only for 15, 21, 27, 33, 48, and 60 nodes. Note that, *MPFE* of *APA* and *GA* are also visible in the Fig. 3.

Fig. 3 shows that APA provides a Pareto set with maximum error (*MPFE*) of around 1% for these networks. Comparing the results of APA and GA, the proposed APA produces much more accurate results than GA in greatly shorter time. Moreover, the minimum, maximum, and median values shown in the figure for APA are smaller than the corresponding values of GA. It shows that APA's Pareto points better spread alongside the reference Pareto front compared to GA even in a twentieth time of GA's execution time. Increasing the number of clusters in the network causes more gap between the performance of APA and GA, which shows the strength of APA compared to GA specifically for larger networks.

To explore the effect of r (closeness parameter) used in APA, we also try $r = 10^{-4}$ and report the results of APA for the same networks in Table III, and the distribution of *PFE* in Fig. 4. Comparing the results in Fig. 4, Table III and Fig. 3 shows that the number of Pareto points and the execution time of the algorithm decreases when r increases, while the error and the number of outliers increases as we expected.

As large scale networks, the tree-cluster networks with 50, 100, and 150 nodes are considered with the same QoS constraints that are set in the previous setups. We assume that these nodes are randomly distributed in a square area with a density of 0.02 node per square meter. Then RPL algorithm is run to establish a cluster-tree network, in a way that each node can accept at most 5 children. Pareto points of these networks are derived using APA and GA with execution time of 20 times the APA's execution time (GA_{20}). For this setup, we skip the production of the reference Pareto points since IPCA fails due to limitation in time and memory. So, the reference Pareto set is not available for this setup. However, it is common in Pareto algebra that Pareto optimal set is unknown in many cases, and an artificial reference set or desired set is specified. Here, we define an artificial reference set as the Pareto optimal of the union of Pareto optimal set of APA and GA. Accordingly, the derived Pareto optimal sets for three large networks are compared with the artificial reference Pareto front, and the results are reported in Table IV. Results show that

TABLE III
PERFORMANCE COMPARISON OF APA WITH $r = 10^{-4}$ AND $r = 10^{-6}$

	$r = 10^{-4}$		$r = 10^{-6}$	
N	$\|APA\|$	t_{APA}	$\|APA\|$	t_{APA}
15	304	14	660	15
21	306	20	661	22
27	310	26	671	27
33	336	32	742	36
48	615	44	1554	67
60	618	55	1567	87

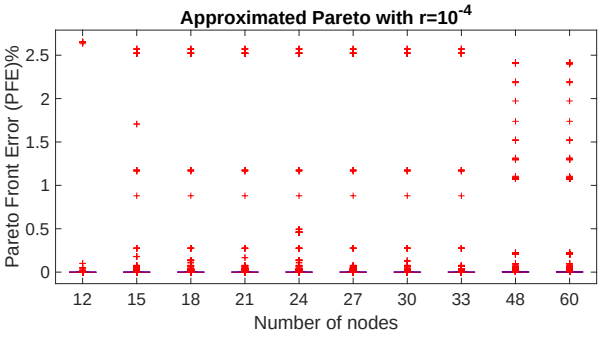

Fig. 4. Distribution of *PFE* for APA

APA produces more accurate and diverse Pareto sets than GA. Moreover, for the networks with 100 and 150 nodes, Pareto points of APA remains in reference Pareto sets that means APA's Pareto set completely dominates the GA's Pareto set.

VII. CONCLUSION

In this paper, an approximated Pareto algorithm is presented to produce a subset of Pareto points for large-scale TSCH network in a short time. Feasibility of incremental Pareto combination algorithm is studied for large networks. It is shown that the incremental Pareto combination algorithm is still not scalable enough for large networks. An approximated Pareto algorithm is proposed to decrease the execution time and complexity to overcome time and memory limitations, which are critical factors in run-time optimization platforms. Comparing the results of the proposed approximated Pareto algorithm with true Pareto for the networks of less than 60 nodes shows that our algorithm produces the Pareto optimal sets with error less than 1% in the order of seconds. Furthermore, the results of the proposed algorithm are compared to the results of the genetic algorithm for different sizes of the network. It is shown that the genetic algorithm during the 20 times the approximated Pareto's execution time produces the Pareto optimal sets at least 10 times less accurate and less diverse than our method for the networks of less than 60 nodes. The results for large networks also show the strength of our proposed algorithm, which provides many Pareto points compared to the genetic algorithm in a twentieth times the genetic algorithm's running time, and those Pareto points completely dominate Pareto points of the genetic algorithm.

VIII. ACKNOWLEDGMENT

This work was supported by the SCOTT European project (www.scott-project.eu) that has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737422.

TABLE IV
PERFORMANCE COMPARISON OF APA (WITH $r = 10^{-2}$) AND GA FOR LARGE NETWORKS

N	$\|S\|$	$\|APA\|$	$\|GA\|_{20}$	$MPFE_{APA}\%$	$MPFE_{GA_{20}}\%$	$t_{APA}(s)$
50	82	80	16	0.008	0.14	50
100	205	205	77	0	0.104	332
150	230	230	42	0	0.175	582

S is the artificial reference Pareto set
$N_{CH} = N/5$

REFERENCES

[1] IEEE Standard for Low-Rate Wireless Networks. *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pages 1–709, April 2016.

[2] H. Dakdouk, E. Tarazona, R. Alami, R. Féraud, G. Z. Papadopoulos, and P. Maillé. Reinforcement Learning Techniques for Optimized Channel Hopping in IEEE 802.15.4-TSCH Networks. In *Proceedings of the 21st ACM Int'l Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWIM 18, page 99107, New York, NY, USA, 2018.

[3] D. De Guglielmo, B. Al Nahas, S. Duquennoy, T. Voigt, and G. Anastasi. Analysis and experimental evaluation of IEEE 802.15.4e TSCH CSMA-CA algorithm. *IEEE Trans. on Vehicular Technology*, 66(2):1573–1588, 2017.

[4] L. Di Puglia Pugliese, D. Zorbas, F. Guerriero, and C. Douligeris. Optimal routing approaches for IEEE 802.15. 4 TSCH networks. *Trans. on Emerging Telecommunications Technologies*, 30(3):e3538, 2019.

[5] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne. Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, SenSys '15, pages 337–350, New York, NY, USA, 2015.

[6] M. Geilen, T. Basten, B. Theelen, and R. Otten. An algebra of Pareto points. In *Fifth International Conference on Application of Concurrency to System Design (ACSD'05)*, pages 88–97, June 2005.

[7] H. Hajizadeh, M. Nabi, R. Tavakoli, and K. Goossens. A Scalable and Fast Model for Performance Analysis of IEEE 802.15.4 TSCH Networks. In *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–7, 2019.

[8] S. Kharb and A. Singhrova. Slot-frame Length Optimization using Hill Climbing for Energy Efficient TSCH Network. *Procedia Computer Science*, 132:541 – 550, 2018. Int'l Conf. on Computational Intelligence and Data Science.

[9] T. Okabe, Y. Jin, and B. Sendhoff. A critical survey of performance indices for multi-objective optimisation. In *the 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, volume 2, pages 878–885 Vol.2, Dec 2003.

[10] R. Hoes and T. Basten and CK. Tham and M. Geilen and H. Corporaal. Analysing QoS trade-offs in wireless sensor networks. In *Proceedings of the 10th International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems, MSWiM*, 2007.

[11] J. Shi and M. Sha. Parameter self-configuration and self-adaptation in industrial wireless sensor-actuator networks. In *IEEE INFOCOM 2019 - IEEE Conf. on Computer Communications*, pages 658–666, 2019.

[12] T. Winter and P. Thubert and A. Brandt and J. Hui and R. Kelsey and P. Levis and Kristofer S. J. Pister and R. Struik and JP. Vasseur and R. Alexander. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. *RFC*, 6550:1–157, 2012.

[13] N. Taheri Javan, M. Sabaei, and V. Hakami. IEEE 802.15.4.e TSCH-Based Scheduling for Throughput Optimization: A Combinatorial Multi-Armed Bandit Approach. *IEEE Sensors Journal*, 20(1):525–537, Jan 2020.

[14] T. Watteyne, J. Weiss, L. Doherty, and J. Simon. Industrial IEEE 802.15.4e networks: Performance and trade-offs. In *2015 IEEE Int'l Conf. on Communications (ICC)*, pages 604–609, June 2015.

[15] B. zceylan, B. nl, and B. Baykal. An energy efficient optimum shared cell scheduling for TSCH networks. In *2017 IEEE 13th Int'l Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–8, Oct 2017.

[16] M. Zlmmerling, F. Ferrari, L. Mottola, T. Voigt, and L. Thiele. pTUNES: Runtime parameter adaptation for low-power MAC protocols. In *2012 ACM/IEEE 11th Int'l Conf. on Information Processing in Sensor Networks (IPSN)*, pages 173–184, 2012.