Flow/Interface Association for Multi-connectivity in Heterogeneous Wireless Networks: e-Health case

Mohamed Abdelkrim Senouci*[a], Hadj Senouci[b], Mustapha Reda Senouci[c], Nasim Ferdosian[d], Abdelhamid Mellouk[a]

[a] *LiSSi Lab., Université Paris-Est, Vitry sur Seine, France.*
[b] *Algérie Télécom., Saida, Algeria.*
[c] *Distributed and Complex Systems Lab., Ecole Militaire Polytechnique, Algiers, Algeria.*
[d] *EURECOM, Campus SophiaTech, Biot, France.*

Abstract

A heterogeneous wireless environment is composed of different radio access technologies (RATs) including IEEE standards (e.g., Wi-Fi) and 3GPP (e.g., 2G, 3G, 4G, 5G), each with different coverage and different capacity to cater for diverse service requirements. In such an environment, multiple access networks could be available for a mobile user. Each user can run multiple applications that have particular Quality of Service (QoS) requirements. Hence, it would be beneficial for a multi-interface terminal to simultaneously use two or more interfaces to gain in performance. However, using multiple networks simultaneously may consume more energy than using only one interface. Therefore, energy consumption should be considered as a criterion in the Flow/Interface Association (FIA). This paper presents a novel method called Smart Tabu Search (STS) that takes into account network conditions, the monetary cost of the network, QoS requirements of the applications, user preferences, and energy consumption of the mobile device to select the optimal FIA that achieves the best trade-off among all the considered criteria. We validate our proposal using both simulations and testbed experiments.

Keywords: Heterogeneous Wireless Networks, Network Interface Selection, Flow/Interface Association, Multi-connectivity, Stochastic Optimization, e-Health.

Nomenclature

2G Second Generation Cellular Technology

3G Third Generation Cellular Technology

3GPP Third Generation Partnership Project

4G Fourth Generation Cellular Technology

5G Fifth Generation Cellular Technology

CF Clever Diversification

ECG Electrocardiogram

FIA Flow/Interface Association

GAP Generalized Assignment Problem

HWN Heterogeneous Wireless Network

LTE/LTE-A Long Term Evolution/LTE-Advanced

MADM Multi-Attribute Decision Making

MDP Markov Decision Process

NIS Network Interface Selection

OTS TS with Oriented Diversification

P% Global optimum probability

RAT Radio Access Technology

SA Simulated Annealing

SCTP Stream Control Transmission Protocol

STS Smart Tabu Search

TOPSIS Technique for Order Preference by Similarity to Ideal Solution

TS Tabu Search

UD Utility Deficit

UMTS Universal Mobile Telecommunications System

WiMAX Worldwide Interoperability for Microwave Access

1. Introduction

The wireless communication networks are different in terms of coverage, cost, and capacities to cater for diverse service needs. Mobile terminals with multiple access interfaces are expected to be always best connected [1], where the multi-interface terminal ranks the Radio Access Technologies (RATs) and selects the one that maximizes its interests and meets the service requirements anywhere at any time, which is known as Network Interface Selection (NIS) [2].

Once a terminal is equipped with several interfaces, called multi-homed terminal, it becomes possible to use the various available networks by associating each data flow with its appropriate network, according to multiple criteria such as network conditions, terminal capacities, user preferences, and application requirements. Associating each application flow with a suitable network interface is considered as a particular case of the NIS and is called Flow/Interface Association (FIA) [3]. Therefore, the FIA allows each multi-homed terminal to spread its flows among the available network interfaces, instead of connecting to only one network interface at a time.

On the one hand, each terminal can simultaneously run different types of applications, which lead to the diverse flows with different Quality of Service (QoS) requirements in terms of throughput, delay, jitter, and packet loss. On the other hand, a multi-homed terminal is equipped with several interfaces; hence, it is possible to simultaneously use various available network interfaces and not simply to switch from a single network to another one. In this context, it would be advantageous for the multi-homed terminal to associate each flow with a specific network interface that meets all its requirements. Many association solutions can be considered as candidate solutions. The objective is to select the association solution that best maximizes the total terminal utility. In other words, the objective is to select the association that best satisfies all QoS requirements of the flows at the least possible cost and efficient use of energy.

In this paper, we tackle the FIA problem, where a multi-homed terminal runs several applications with different QoS requirements. Each application flow is associated with an appropriate network interface. The main objective of our proposed approach is to meet the specific QoS requirements of each flow, while maximizing the overall utility of the terminal; consequently, the association that will be selected is the one with the maximum global utility.

The rest of this paper is organized as follows: Section 2 presents the related work. In Section 3, we discuss about the FIA concept and we present the model description. Section 4 presents a comparative study of two random search approaches widely applied to stochastic optimization problems in the context of the FIA, namely Simulated Annealing (SA) and Tabu Search (TS). The proposed scheme is described in Section 5. Section 6 presents and discusses the obtained results of the simulation. The design and implementation of the testbed are presented in Section 7. In Section 8, we describe the testbed setup, execution and the obtained testbed results. Finally, Section 9 concludes the paper and presents the future work.

2. Related work

In heterogeneous wireless networks (HWNs), the multi-homed mobile terminals rank the network interfaces and select the best one or simultaneously use the various available network interfaces. To meet the always best connected requirements, studies on NIS have used different mathematical theories including utility function [4, 5], cost function [6], Multiple Attribute Decision Making (MADM) [7, 2, 8, 9], Markov chain [10, 11, 12, 13], fuzzy logic [14, 15], game theory [16, 17], and belief functions theory [18].

Authors in [5] used a utility function for NIS, where the main idea is to evaluate the utilities of all criteria and combine them to obtain one total utility for each candidate network, and the one with the highest overall utility value is selected as the best one. In [6], a cost function is used to measure the cost caused by the use of each candidate network, and the one with the lowest cost is considered the best one. Normally, the cost of an alternative can be seen as the inverse of its utility, where the form of this inversion depends on the way the attributes were combined.

In [2], the authors used Technique for Order Preference by Similarity to Ideal Solution (TOPSIS), a MADM approach for the NIS problem and considered a set of attributes in the decision making process. The value of each attribute is normalized and assigned a weight, the best and the worst values are found for each attribute, and then a score function is computed based on the distances for both best and worst cases. The network with the highest score value is selected. The proposed approach efficiently tackles the rank reversal problem in TOPSIS by using new normalization techniques. Authors in [9] combined TOPSIS with the utility function to tackle the rank reversal problem and enhance the ranking quality by considering the application's and/or the user's requirements. TOPSIS was used to rank the candidate networks based on their scores with the highest being the best, and the utility function was used to compute the normalized value of each parameter.

In Markov chains, such as Markov Decision Process (MDP)-based schemes [11, 12, 13] the objective is to maximize the expected total reward of a connection to optimize the NIS decisions. The algorithms use two types of functions: a link reward function and a cost function. The Link reward function is associated with the QoS provided by the selected network to the mobile connection, and the cost function is associated with the rerouting operation and signaling load incurred when a vertical handover occurs.

Authors in [14] proposed a fuzzy logic based-scheme for the NIS problem without combining it with any other theories and is adaptable to the changes of network and traffic, and integrates the uncertain and conflicting metrics to make a perceivable decision inexpensively. Many approaches have been proposed to solve the NIS problem using fuzzy logic with different ways. Some approaches use the fuzzy logic as the core of the NIS system [14], and others use it as the fuzzy MADM [15], while some use it with recursion procedure [19].

The authors in [17] modeled the NIS problem into a non-cooperative game belonging to the class of congestion games between selfish users. In this game,

the users take their actions on selecting one network among the available ones. The cost of each user depends on the congestion of the selected network. This game becomes a problem in which all the users try to choose the network with minimum cost.

In [18], a belief function theory (BFT) was used for NIS problem. All candidate networks are judged with different masses (degrees of belief), where the belief reflects the satisfactory extents for each candidate network from the viewpoint of this specific parameter. In the case of multi-criteria, one network will have multiple judgments. To obtain one judgment that reflects a comprehensive satisfactory rate for that network, all the masses are combined together using conflict-aware combination rules. The candidate network with the highest combined mass is selected as the best one.

The most of the NIS approaches in the literature do not consider the possibility of using two or more network interfaces simultaneously. In [3], a metaheuristic approach based on TS that considers the possibility of a mobile terminal to use multiple network interfaces simultaneously, considering the problem as an optimization problem, has been proposed. Using the simulation, the authors exhibit that the standard algorithm has poor performance to find the global optimal solution. Further analysis shows that the algorithm repeatedly gets entrapped in the local optimum which prevents it from exploring other solutions. The random diversification, used in the standard Tabu implementation, is defined as the source of the problem, and a new method named "Oriented diversification" is proposed to remedy the issue. This method looks up the Tabu list for an application that has not been tested on one of the available networks and forces that selection in the newly generated association, thus preventing the diversification step from producing an association that is previously explored by the algorithm and overcome the entrapment issue. Comparisons through simulation show a significant improvement in terms of results and computational time in favor of the proposed algorithm.

This paper contributes to the body of knowledge in this area by proposing a novel method that takes into account network conditions, the monetary cost of the network, QoS requirements of the application, user preferences, and energy consumption of the mobile device to select the optimal FIA which achieves the best trade-off among all the considered criteria.

3. Flow/Interface Association

Once a terminal is equipped with several interfaces, it becomes possible to use two or more interfaces simultaneously. The terminal in a zone covered by HWNs can discover different RATs with different characteristics (e.g., UMTS, LTE, WiFi, WiMAX, etc.). The characteristics of RATs such as the WiMAX and UMTS are dynamic, which will affect their ability to support a service optimally. The wireless network technologies are developed to fulfill the specific requirements of applications, for example LTE-A is developed to fulfill the multimedia services such as VoIP, HD video streaming, interactive video gaming, etc.). In case the multi-interface terminal runs several applications at the

same time, heterogeneous data flows with different requirements in terms of QoS need to be fulfilled. In this case, it would be advantageous to provide each data flow with an appropriate network interface that meets its requirements while maximizing the global terminal utility.

Example: A simple scenario is presented in Figure 1, where three applications (Telepresence, E-mail, and video Streaming) are run in a multi-homed terminal which is equipped with three network interfaces (LTE, 802.11ac, and 802.11b). In real-time critical application, jitter, packet loss, and low latency are mandatory. The Telepresence flow will be associated with LTE as it has low values of these parameters (for simplicity, we did not consider the cost criterion in this example). For a background application even a network with low performance will suffice, then the E-mail flow will be associated with 802.11b. Finally, the streaming flow will be associated with 802.11ac as it has a high throughput compared to 802.11b and low energy consumption compared to LTE.

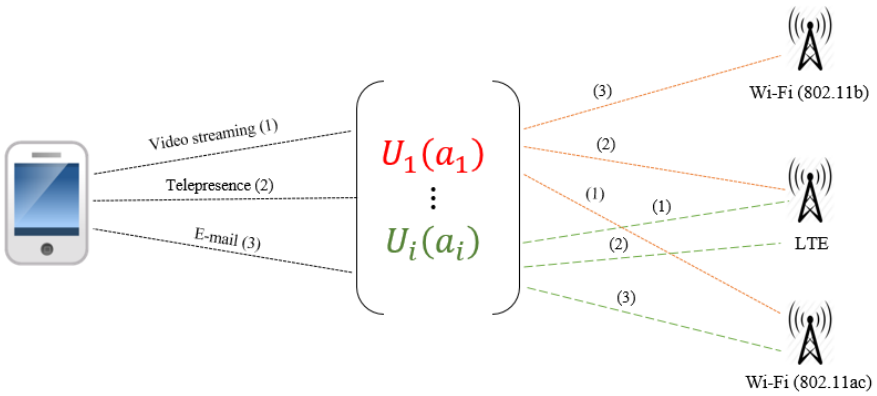

Figure 1: Flow/Interface Association model.

3.1. Model description

The FIA is a generalization of an assignment problem in which there are n agents and m tasks. Any agent can be assigned to perform any task, incurring some *cost* and $profit$ that may vary depending on the agent-task assignment. Moreover, each agent has a *budget* and the sum of the costs of the tasks assigned to it cannot exceed this budget. It is required to find an assignment in which all the agents do not exceed their budget and the total profit of the assignments is maximized.

Mathematically a Generalized Assignment Problem (GAP) can be formulated as an integer program:

$$Max \sum_{i=1}^{n} \sum_{j=1}^{m} p_{i,j} x_{i,j}$$

$$s.t. \sum_{j=1}^{m} c_{i,j} x_{i,j} <= t_i, \; where \, i = 1, ..., n;$$

$$\sum_{i=1}^{n} x_{i,j} = 1, \; where \, j = 1, ..., m;$$

$$x_{i,j} \in \{0, 1\}, \; where \, i = 1, ..., n \, and \, j = 1, ..., m;$$

n Agent: $I_1 ... I_n$, m Task: $F_1 ... F_m$, p: Utility, c: Cost, t: Budget.

A feasible solution is a solution in which for each Agent the total cost of the assigned Tasks is at most t_i. The solution's utility is the sum of utilities for each task-agent assignment. The goal is to find a feasible solution with maximum utility, which is exactly what we are trying to solve in the FIA problem.

Consider n network interfaces and m flows, where $I = \{I_i, i = 1, 2, ..., n\}$ and $F = \{F_i, i = 1, 2, ..., m\}$. Thus, the set of possible association can be represented by n^m, where $A = \{A_i, i = 1, 2, ..., n^m\}$.

Each association $a_i \in A$ allows the set m to be associated to the set n, so each $f_j \in F$ is associated to only one $i_k \in I$ and each i_k may take from 0 to m flows. For example, a solution a_i provides the association of f_1 to i_1, f_2 and f_3 to i_3, ..., and f_m to i_n.

The utility for flow f_j associated to interface i_k is calculated as follows:

$$U_{a_{i,j}} = \alpha . Q_{k,j} + \beta . C_{k,j} + \gamma . E_{k,j},$$

where $Q_{k,j}$, $C_{k,j}$ and $E_{k,j}$ represent the QoS, cost and energy consumption utilities of assigning f_j to i_k determined by a_i, and α, β and γ are their weights respectively.

As illustrated in Figure 1, for each association a_i, a total utility U_i indicated by $U_i(a_i)$ is calculated as follows:

$$U_i(a_i) = \sum_{j=1}^{m} U_{a_{i,j}}.$$

The goal is to maximize the overall utility of the terminal, consequently, the association that will be selected is the one with the maximum global utility.

$$max \, U_i(a_i) \quad i = 1, ..., n^m. \tag{1}$$

The utility function of an interface represents the satisfaction level of flow(s) assigned to that network interface. The association solution with the maximum total utility value can be considered as the best solution only if it is a feasible solution. The latter means that each network interface should satisfy all the requirements of the flows associated to it. Otherwise, the solution will not be feasible (Section 4.1 further explains this part). The GAP/FIA is known to be NP-hard [20]. Thus, aiming for an exact solution is computationally taxing and time consuming. Moreover, in the case of mobile terminals, both of

these resources are very scarce. Therefore, we used Metaheuristic algorithms to address this problem.

In the following, we provide a simulation-based comparative study of two random search algorithms, widely applied to stochastic optimization problems, including SA and TS in the context of this study.

4. Algorithms comparison

We consider two approaches, namely Simulated Annealing from the physical class, and Tabu Search from the stochastic class. We also consider the Brute-force algorithm, as a reference in the comparative study, as it is the one that can find the global optimum. Python 2.7 was used for the implementation and simulations. The simulation results show the advantages and disadvantages of the two algorithms in the context of FIA. The motivation of our proposal is to remedy the shortcomings. It is worth mentioning that we did not consider genetic algorithms because there is no genetic relationship in our context, where the genetic algorithm intents to make new generations of the current population with characteristics related to reproduction. In the FIA context, two association solutions do not produce a new association solution that receives genetically their characteristics. Therefore, the genetic algorithm cannot be a solution for the FIA problem.

4.1. Simulation setup

In the simulation, we consider three classes containing different types of flows. For each flow, the different QoS settings are defined with specific utility functions based on the analysis in [21, 22] as follows:

- Class 1 "Critical" covers Telepresence and PCM VoIP (G.711), and uses Real-Time.
- Class 2 "Greedy" comprises P2P, FTP, and Downloading, and uses Scavenger.
- Class 3 "Elastic" includes audio and video streaming, and uses Best Effort.

Four QoS criteria, including Bandwidth in [Mbps], Jitter in [ms], Packet Loss in [PER], and Latency in [ms] are considered to measure the interface utility, where their requirements are defined according to the type of the corresponding class.

The mutil-interface terminal is supplied with I network interfaces, where ($I \geq 2$), and one RAT may cover multiple network interfaces. One network interface can take F flow(s), where ($F \geq 0$). Six wireless communication networks are considered in the simulation (WiFi 802.11b, 802.11a, 802.11n, 2G, 3G, and 4G) with six parameters including four QoS parameters (Bandwidth [Mbps], Packet Loss [PER], Jitter [ms], Latency [ms]), Energy Consumption [mW/Mbit], and Cost [cent/MB]. For each RAT, the value of each parameter

can be obtained from a predefined interval, so the same RAT can be generated with different specs.

The QoS, energy consumption, and cost variables are weighted according to user preferences where the sum of the three values (weights) is equal to one. The variables' weights represent the relative importance of these variables. The QoS, energy consumption and cost represent the weights of flow, user and terminal requirements respectively.

Interfaces that accommodate two or more flows will allocate resources according to the combined requirements of flows. If only one interface does not meet one criterion's requirements, then the entire association solution is considered invalid. There are two ways to handle the invalid association solution.

a) The valid and invalid association solutions are both tracked by the algorithm. The utility value may not be appropriate for choosing an association solution since some invalid association solutions may have higher utility values than the valid ones. This method will provide the user with two choices, and since two association solutions are maintained, then the invalid association solution can be chosen only if it has a very high utility value, but with slightly higher memory footprint and CPU.

b) The number of violations for an interface is computed and used to normalize the utility, so the invalid association solution will always have a lower value than the valid one. With this approach, the best association solution can be chosen according to the utility value, but the user is not involved in the decision making.

4.2. Simulation scenarios

We consider two scenarios in the simulation. First scenario involves six flows including, audio streaming, video streaming, P2P, Telepresence, and 2x VOIP and a multi-interface terminal with four network interfaces, including 802.11g, 802.11n, 802.11ac, and LTE. There are 4^6 (4096) possible association solutions referred to as nodes. The objective of the algorithms is to find the global optimum by exploring the preset number of nodes.

In the second scenario, we consider ten flows, including 2x audio streaming, 1x video streaming, 2x P2P, 2x Telepresence, and 2x VOIP and five network interfaces, including 802.11g, 802.11n, 802.11ac, UMTS and LTE. In this scenario we have 5^{10} (9765625) possible association solutions. We use the Brute-force algorithm as a reference against which the algorithms' performances are compared according to three criteria as shown in Table 1

- Global optimum probability (P%) [0, 1]: it represents the probability of an algorithm to find the global optimum.
- Utility Deficit (UD) [0, 1]: it measures how far are the association solutions from the global optimum. If no association solution is found then the UD value is equal to 1, the closer the solution gets to global optimum, the lower the values of UD gets, until it reaches 0 which means the solution found is a global optimum.

- Speedup (SP): it represents the speedup gain of an algorithm compared to Brute-force search.

All results illustrated below are an average of the execution of the algorithm N times with the same set of network interfaces but with different specs.

The first scenario is executed 100 times, and the second scenario is executed 25 times. The average running time of both scenarios using the Brute-force algorithm is 0.5326 seconds and 1776.8030 seconds, respectively.

As a stop criterion, it is possible, for example, to set a maximum number of iterations without improvement, fix a time limit or define a certain number of iterations after which the search must stop. However, we chose the number of nodes allowed to be explored as a stopping criterion, since it is more precise than the number of iterations, as during each iteration, different algorithms will explore a different number of solutions.

4.3. Simulation results

Table 1: Scenario 1–Tabu Search & Simulated Annealing Results

Nodes	Tabu Search			Simulated Annealing		
	P%	UD	SP	P%	UD	SP
10	0.0	0.93167	109.8	0.0	0.93167	110.8
100	0.04	0.31232	18.8	0.01	0.55582	19.1
200	0.11	0.17683	10.3	0.02	0.36657	9.7
400	0.16	0.15604	5.4	0.07	0.20126	5.4
1K	0.3	0.03866	2.4	0.11	0.08183	2.4
2K	0.46	0.02642	1.3	0.33	0.02357	1.2

As shown in Table 1, more the number of explored nodes increases, more $P\%$ and UD improve while speedup diminishes; since exploring more nodes involves more execution time but with better chance to converge to the global optimum.

The second remark is that TS outperforms SA. However, it should be noted that even by exploring about 50% of all possibilities both algorithms deliver low-quality solutions with very modest speedup. To enhance further their performance, we propose two minor modifications (results are presented in Table 2):

- Neighborhood by displacement: rather than considering a randomly-generated neighborhood, we pick the current solution and assign each flow to all available interfaces one at a time. For illustration purposes, let us assume two interfaces, three flows, and the current solution = [1, 1, 2]. This indicates that flows 1, 2, and 3 are assigned to interfaces 1, 1, and 2, respectively. Neighborhood = [[2, 1, 2] [1, 2, 2] [1, 1, 1]].

- Caching: SA and TS (and especially SA) will revisit many solutions, hence their neighborhood. If some of the neighborhood results are cached, it should improve the run-time complexity of both algorithms.

Table 2: Scenario 1–Results for TS & SA with minor modifications

Nodes	Tabu Search			Simulated Annealing		
	P%	UD	SP	P%	UD	SP
10	0.00	0.00000	138.9	0.00	0.00000	138.7
100	0.65	0.18419	30.3	0.67	0.13411	29.8
200	0.88	0.03139	14.6	0.98	0.00029	24.2
1K	0.94	0.00086	2.9	0.98	0.00029	22.7
2K	0.97	0.00043	1.4	0.98	0.00029	19.5
20k	1.0	0.0	0.13	0.98	0.00028	3.1

The enhancement in P% and UD is apparent with SA reaches its peak faster than TS at around 200 nodes. After that, SA gets stuck in local optimum, which is typical of SA. Indeed, at lower temperatures, SA tends to seize exploration and hold to the current best solution. In the other hand, TS manages to find all global optimum, despite the fact that the improvement is slower as compared to that of the SA. These results confirm two points:

- The outcome is heavily influenced by the neighborhood generation method;
- 100% success rate is expensive.

Concerning the caching, it essentially helps SA due to the nature of this latter. Indeed, SA lacks a procedure to avoid looping over the same search space; therefore, reexamining previously explored solutions is likely. In the other hand, TS did not really benefit from the caching. This could be explained by the fact that TS avoids revisiting solutions by conserving the Tabu list. The minor enhancement, noticed in the obtained results, is due to the cases where the revisited nodes were produced from solutions that are either not Tabu or were Tabu but removed during the Tabu list update. As the benefit of these slight modifications is obvious, and in order to provide fair results, all competitive algorithms considered in the performance evaluation will undergo these modifications, including Tabu with oriented diversification [3].

For small-scale problems, both algorithms' performance improved; however, for large-scale problems (e.g., the second scenario) algorithms' performance decreases (see Table 3). The P% is directly proportional to the number of solutions explored, though the larger the set of nodes is, the more time is needed, which is ineffective in the context of FIA. In fact, the end-user will endeavor to achieve the maximum utility, but only as long as it is attainable during a specific time frame. The dynamic nature of mobile networks is another factor to be considered. Obviously, in order to be useful, the results must be provided in real-time, as it is useless to exploit the results while some networks are no longer accessible and/or other new networks are available.

Table 3: Scenario 2–Results for TS & SA

Nodes	Standard Tabu			Simulated Annealing		
	P%	UD	SP	P%	UD	SP
10	0.0	1.00000	109105.8	0.0	1.00000	112664.5
100	0.0	1.00000	38136.4	0.0	1.00000	37176.4
200	0.0	0.72653	21110.5	0.0	0.96108	23726.3
1k	0.72	0.08222	5931.3	0.84	0.00199	14210.1
10k	0.72	0.00247	570.9	0.84	0.00199	13799.1
50k	0.96	0.00036	90.5	0.84	0.00199	9903.5

5. Our proposal

In this section, we seek to remedy the shortcomings indicated previously. We chose TS as the main algorithm, although it has lower performance compared to SA, and that is for two reasons: (i) TS does not seize exploration, unlike SA whose exploration relies on the temperature variable which eventually becomes low and the algorithm remains steady, and (ii) TS can employ different types of memory (short, intermediate, and long-term) to bias move towards promising areas of the search space or promote a general diversity. To enhance the performance of TS, we propose a new method called Smart Tabu Search (STS) that improves the original TS on three points, the Clever Diversification, Approximate Initial Solution, and Oscillation.

5.1. Clever Diversification

TS uses random diversification to escape from getting stuck on a local optimum, when there is a slump in the solution improvement, the algorithm produces a random candidate and explores its neighborhood for a better solution. If one is found, TS updates its state and continues from there, otherwise, new diversification is executed. Two defects can be identified in the current process.

- For a demonstration, let I equals the number of interfaces and F the number of flows. S is the search space, where $S = I^F$, and Ns is neighborhood size, where $Ns = (I-1)*F$ (using the displacement technique). It is clear that as the search space gets larger, the number of neighborhoods $N = \frac{S}{Ns}$ gets bigger, which means the chance to produce a candidate, whose neighborhood contains the global optimum, gets extremely small with a large S.
- The process being totally random, means it will produce candidates that are far from being optimal and the algorithm will spend a lot of time exploring them.

To address these issues, we propose a clever diversification (CF). In CF, the search space from which the candidate is chosen is kept smaller, although it still grows with S but with a lower rate. This will increase the chance to

stumble upon a neighborhood with the global optimum. Second, the candidates produced by CF will contain interfaces that are likely to be part of the global optimum.

To explain how CF works, let us assume that we have four flows and three interfaces. The procedure to generate a candidate includes two steps:

1. The first step aims to reduce the search space by using the Algorithm 1. The output of the algorithm is a matrix M, where columns represent flows and rows represent available network interfaces for each flow. For example, in the matrix below, flow 1 can select interface 1 or 2, and flow 2 can be associated only to interface 1.

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & - & 2 & - \\ - & - & 3 & - \end{bmatrix}$$

2. The second step generates a candidate, where network interfaces are randomly assigned to flows as shown in the matrix M. If the resulted candidate is Tabu, then the process is repeated for a maximum of n times ($n > 0$). If no candidate is generated then the random diversification is executed.

5.2. Approximate Initial Solution

TS begins by a randomly generated candidate as an initial solution and starts improving it. Sometimes the provided candidate can be far from the global optimum and/or invalid, thus a considerable amount of time will be spent to improve it. Instead, an approximate solution with high utility and valid (if possible) is provided to the algorithm, hence TS has a higher chance to stumble upon the global optimum quickly.

To create an approximate solution, we find the interface with the highest utility for each flow and assign it to the latter, then update the interface's expandable resources (e.g. bandwidth, buffer, etc.) to reflect the allocation. The procedure requires $I * F$ evaluations. In addition to the simplicity of the technique, it significantly improves the performance as it is demonstrated in the next section.

5.3. Oscillation

Normally, TS only accepts solutions with better or equal utilities. This behavior makes the algorithm greedy and it might get stuck on a local optimum. By introducing *diversification*, we provide the algorithm with random candidates that lead to the global optimum and mitigate the greedy behavior. But this holds true only if the new candidates or their neighborhoods have solutions with a better utility than the current best. This process is repeated when there is no improvements and it gets longer when the current solution gets closer to

Algorithm 1 Reduce Diversification Space

Require: M empty matrix

for f in Flows **do**
  for i in Interfaces **do**
    Calculate the utility U_{fi} using equations [9]:
    for upward attributes (e.g., Throughput):

$$f(x) = L - (L - b)e^{(-k(x-a))} \tag{2}$$

k is computed using the following equation:

$$k = -\ln(1-p)/(\textit{Target point} - a),\ 0 < p < 1 \tag{3}$$

    for downward attributes (e.g., Latency)

$$f(x) = L - e^{(k-(x-a))} \tag{4}$$

    for monetary cost:

$$\begin{cases} f(x) = 1 - x/u,\ x \geq u \\ f(x) = 0,\ x > u \end{cases} \tag{5}$$

  end for
  Calculate the mean m and std. deviation sd of U_{fi}
  SD_{only} = False
  for i in Interfaces **do**
    if $\neg SD_{only}$ **then**
      if $U_{fi} \geq m + sd$ **then**
        Clear $M[f]$
        SD_{only} = True
        Add i to $M[f]$
      else if $m - sd \geq U_{fi} \geq m + sd$ **then**
        Add i to $M[f]$
      else
        Discard i
      end if
    else if $U_{fi} \geq m + sd$ **then**
      Add i to $M[f]$
    end if
  end for
end for

the global optimum, since the number of solutions better than the current best utility diminishes.

To explain, consider the following example. Assume the present solution is second to the global optimum, and to reach the optimum, two flows have to change their interfaces. Generating the neighborhood by displacement only alters one flow each time. That means the algorithm cannot reach the global optimum, because TS would not explore solutions with inferior utility (see Table 5).

This problem can be solved by allowing TS to explore and accept sub-optimal solutions. To realize that we use a technique similar to SA's acceptance probability (Equation 6), but contrary to SA, when the number of iterations since the last improvement rises (resp. c is set low) the probability increases. This will render TS to be more stochastic when improvements halt. The probability is reset when a better solution is found, to permit the algorithm to search the vicinity of the current solution.

$$Oscillation_{Pr} + \frac{Last_{improvement}}{c} < random(0,1), \tag{6}$$

where c is a constant, $c \geq 1$.

6. Performance evaluation

The combination of the improvements presented in the previous section and TS lead to a new algorithm called Smart Tabu Search (STS). The new algorithm is compared with the TS with oriented diversification (OTS) [3] running the same scenarios presented above. Results in Tables 4 and 5 highlight the performance gain of STS.

Table 4: OTS *vs.* STS – Scenario 1

N (Nodes)	Oriented Tabu			STS		
	P%	UD	SP	P%	UD	SP
10	0.0	1.00000	140.0	0.99	0.00005	117.4
100	0.65	0.17584	30.9	1.00	0.00000	31.7
200	0.89	0.05146	17.0	1.00	0.00000	14.8
1k	0.95	0.00078	3.1	1.00	0.00000	2.9
2k	0.96	0.00062	1.4	1.00	0.00000	1.4
20k	1.0	0.00000	0.5	1.0	0.00000	0.5

From the results, it is obvious that OTS does not provide a lot of advantages compared to the standard Tabu, and that is because of the way oriented diversification works. For a given input, it takes the $TabuList$ and search for flows that have not yet visited all interfaces, and randomly assigns one of the flows to one of the unvisited interfaces, or resort to random diversification if such a case cannot be found in the TabuList. The approach is futile when the candidate's neighborhood is generated by the displacement method; since the majority of

the results generated by the oriented diversification are already evaluated when exploring the neighborhood. We tried to randomly generate the neighborhood, but OTS performs even poorly compared to the TS, however, there is a particular case where the approach can be helpful, and that is when we meet a situation similar to the example discussed previously in the oscillation section, represented in Table 5. Where OTS can find all global optimum by evaluating 50k nodes, contrary to the STS (without oscillation) where the search halts at 96%, although will still converge albeit very slowly.

As for STS the results are apparent, for the first scenario, the algorithm begins at 99% success rate, thanks to the approximate initial solution, and reaches 100% after evaluating 100 nodes only. This proves that supplying a high utility initial solution yields better results quickly. The second observation is that STS is slightly slower compared to the OTS and that is expectable, since computing the initial solution and limiting the search space for the diversification process takes time; but that is apparent on small problems only, the overhead is negligible as the search space becomes larger. First case of the second scenario shows that STS is capable of solving large problems with affecting its performance, and although the algorithm converges quickly, there are cases where it gets stuck on a local optimum early and ceases to improve. The second case demonstrates the importance of oscillation. By simply allowing the algorithm to accept and explore sub-optimal solutions, we can escape the local optimum.

Table 5: OTS *vs.* STS – Scenario 2

Nodes	Oriented Tabu			STS					
				Case 1: no oscillation			Case 2: oscillation		
	P%	UD	SP	P%	UD	SP	P%	UD	SP
10	0.0	1.0000	96388	0.8	0.0009	118066	0.8	0.0009	115503
100	0.0	1.0000	38319	0.96	0.0001	106503	1.0	0.0000	47135
200	0.0	0.8042	19662	0.96	0.0001	32507	1.0	0.0000	25230
1k	0.76	0.0423	6856	0.96	0.0001	5955			
10k	0.76	0.0423	589	0.96	0.0001	570			
50k	1.0	0.0000	91	0.96	0.0001	100			

To further quantify the real benefit of our approach, we developed a real health monitoring testbed, named *RaspNet*, comprising multiple sensors for medical application. In the following section, we will present the work carried out toward the design, development and implementation of our testbed in the LiSSi laboratory. We will also report the obtained results, supported by the comparison with other works.

7. Design and Implementation of a Real Health Monitoring Testbed

In this section, we present the implementation details of our proposition in an appropriately configured *RaspNet* and discuss the scenarios we have tested. More specifically, an e-Health application is considered to assess the impact

of the selection algorithm on various parameters. Five medical sensors from Libelium [23] are used to capture different biometrics, namely ECG, EMG, GSR, Airflow and a Thermometer. The sensors are wired to an e-Health Sensor Shield (PCB) [24] from the same company which in turns connect to Arduino, Intel Galileo or Raspberry Pi, the latter is used in this study (see Figure 2). This setup (sensor node) can be used to monitor the status of a patient in real-time or to get sensitive data in order to be subsequently analyzed for medical diagnosis.

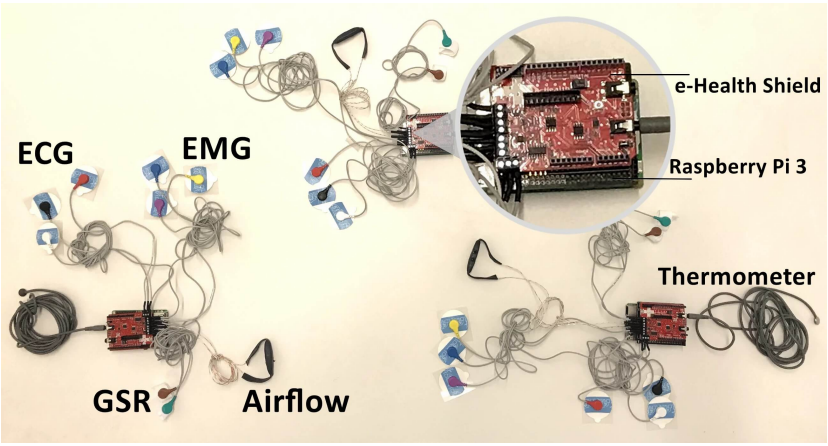

Figure 2: The e-Health platform.

The gathered information is wirelessly sent over LAN using any of the two connectivity options available: Wi-Fi and Bluetooth to another Raspberry Pi (router node) connected to the Internet. The router node has five wireless network interfaces: built-in Wi-Fi and Bluetooth for LAN communications with the sensor node, and 4G, 3G and Wi-Fi dongle with Internet access for data forwarding. The router node's main objective is routing the received data from the sensor node(s) to the server through the appropriate interface according to the application and user requirements using a selection algorithm (see Figure 3).

This entity takes into consideration several pieces of information, such as the monetary cost, the network performance, and etc. Some of these characteristics are collected from the operating system of the device or are hard-coded due to the lack of an appropriate equipment.

The testbed is based on client/server architecture, both of which run on different Linux flavors.

7.1. Client

The client comprises two entities with different tasks. A **sensor node** that captures patient's vital signs using different medical sensors, and a second entity, known as **router node** responsible for routing the captured data via the most suitable network interface.

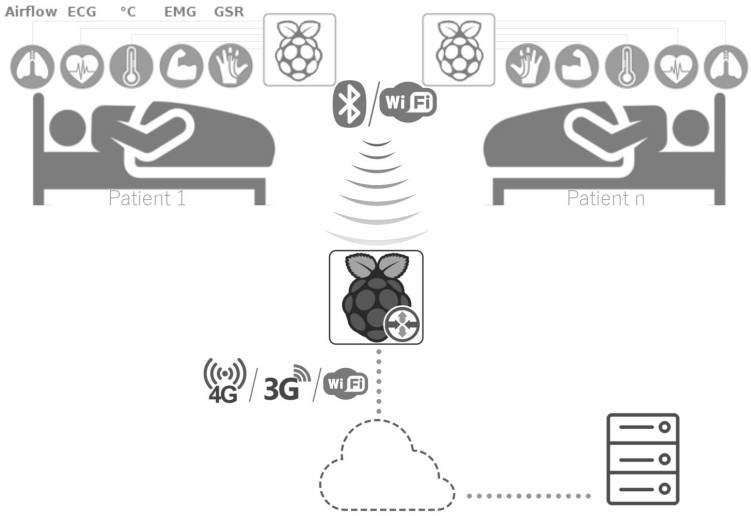

Figure 3: The testbed setup.

7.1.1. Sensor node

The sensor node runs on Raspberry pi 3 with raspbian stretch OS. It contains an e-Health Sensor Shield (PCB) mounted on top of the Raspberry pi 3, and an ECG, EMG, GSR, Airflow, and a Thermometer connected to the e-Health Sensor Shield. Each sensor stores the captured data in a file with real-time updates, then the files are sent over Wi-Fi or Bluetooth in LAN setup to the router node.

7.1.2. Router node

The router node also runs on Raspberry pi 3 with raspbian stretch OS, it has access to the Internet through three different RATs: 4G, 3G, and Wi-Fi. The node's main objective is sending the received data from other sensor nodes using the most suitable network interface according to the flow requirements and user preferences. The implemented system powering the node consists of five separate but complementary modules. The modules objectives are as follows:

1. **Network Probing:** This module collects all the information about the parameters pertaining to NIS, such as the monetary cost, the network performance, the power consumption, the signal strength, speed, GPS position, etc. The type of information collected depends on the application requirements. In this study, the first three set of information are considered.

    - **Power Consumption**: It is difficult to measure accurately. An

estimation could be made based on the wireless Network Interface Card (NIC) transmission power and data rate, but this method does not consider the power consumption when the interface is idle or receiving data. Another approach is to compute the terminal's power consumption over a period of time t (in seconds) with all network interfaces turned off, then doing the same with one interface turned on and operating (sending and receiving). The difference between these two measurements divided by t is the average network interface's power consumption per second. We adopted the second approach, and calculated power consumption for all available wireless NICs, results are depicted in Table 6.

- **Monetary Cost**: Since it is fluctuating and based on the providers' economic policies, the values for each interface are precomputed based on the carrier's data plan and hard-coded.
- **Network Performance**: It refers to the measurement of the overall performance of a service, as seen by the customer. There are many different ways to measure the performance of a network, the following measures are often considered important:
  - **Throughput** is the actual transfer rate for the data.
  - **Latency** is the delay between the sender and the receiver.
  - **Jitter** is the variation in packet delay at the receiver.
  - **Packet Loss** is the number of corrupted packets expressed as a percentage or fraction of the total sent.

There are many tools available to measure one or more of these parameters, some of which are proprietary and specific to vendor applications. The measurement process is typically undertaken by sending a number of probing packets from one system (sender) to another (receiver), when the packets arrive at the receiver, some predefined metrics are computed and reported.

A more accurate method is to use a dedicated software such as Spirent Test Center, JDSU QT600, Netcps, IxChariot, Iperf3, Ttcp, netperf, NetPIPE, Flowgrind or bwping for measuring different parameters.

We have chosen Iperf3 [25], a widely used open-source and cross-platform tool for network performance measurement and tuning. It can produce standardized performance measurements for any network. Iperf3 has client and server functionality, and can create data streams to measure different parameters between the two ends in one or both directions. The data streams can be either TCP, UDP or SCTP. *Throughput* is measured with TCP while UDP is used for *Jitter* and *Packet loss*.

Iperf3 does not measure *Latency*, therefore another tool is required for that. Latency can be either One-Way Delay (OWD), i.e., latency in one direction or in terms of Round-Trip-Time (RTT), which is the length of time it takes for a signal to be sent plus the length of time it takes for

an acknowledgment of that signal to be received. The OWD is more relevant measure than RTT for applications that are primarily producers or consumers of data, since delays between two network nodes are not completely symmetric.

OWAMP [26] is an implementation of the One-Way Active Measurement Protocol ***OWAMP*** as defined by RFC 4656. It is used to determine the OWD. However, OWAMP drawback lies in its difficulty to use, it requires a Network Time Protocol (NTP) synched client and server with at least four peers for the measurements to be accurate. Another method that is easy to accomplish (with a simple ping) but would produce an approximation is to halve the RTT value. The accuracy of such an estimation depends on the delay distribution in both directions: as delays in both directions become more symmetric, the accuracy increases.

The tests are executed in parallel for all available NICs to reduce execution time, but that requires the server's throughput to be greater than all the client NICs' combined for the measurement to be as accurate as possible; otherwise, sequential execution is performed. Our server has enough throughput for a parallel approach. The execution time takes about 6 seconds in total, 3 to determine throughput and another 3 for the remaining parameters.

These tests can throttle the transmission of other data through the Internet connection as they are undertaken, and can cause inflated data charges. Therefore, it stands to reason to run it sparingly or when needed. In this testbed, the module is executed routinely and asynchronously at a predefined interval, but the frequency is subject to change depending on different factors such as mobility and terminal speed. It is also possible to force its execution for certain events, such as:

- Disconnected network,
- NIC is down or up.

QoS, monetary and power consumption values of the used networks are detailed in Table 6. The ones marked as dynamic are for reference only, since they are subject to change and needs to be computed during the program execution. Others are pre-computed and hard coded.

2. **Policies:** They provide information about flow requirements and user preferences to the NIS module. Every flow has to define the required QoS and user preference values. In order to simplify this issue for the less experienced users, we predefined a list of the most usable values as follows.

   - **QoS**
     - Real-time voice (e.g. PCM VoIP)
     - Real-time data (e.g. TelePresence)
     - Best effort voice (e.g. audio streaming)

  - Best effort data (e.g. video streaming)
  - Scavenger (e.g. FTP, P2P or Torrent)
- **User preferences**
  - Performance
  - Cost
  - Energy
  - Balanced

3. **NIS:** This module is responsible for assigning each data flow to an appropriate network interface while considering multiple factors. It takes as input flows requirements, user preferences, and network interfaces measurements. The output is a priority list of the two most suitable networks for each flow as determined by the selection algorithm. Different algorithms will produce different results, but the main goal is to determine the global optimum or at least approach it in case of large search spaces. Five selection algorithms are considered namely, Brute-force, STS, TS, SA, and Random. Brute-force is used as a standard by which other algorithms are measured since it always finds the optimum solution. While the random strategy will simply assign each flow to a network interface randomly, and is meant to exhibit the efficacy of other selection algorithms.

4. **Transmission:** It is implemented as threads, a thread for each data flow. The module objective is to send the data read from a file through the given network interface using TCP socket, if the network is disconnected or down then the second interface is used. If all interfaces are down, the thread status is updated to "disconnected" and communicated to the main program to force a new network probing, followed by a NIS. The selection results are communicated back to the thread, and transmission is resumed. The module will run continuously until all data are sent or discarded by the user (e.g. file deleted). The end of data flow is determined by setting a timer on the given file, if the timer runs out and no write event is reported by the inotify API then the flow is considered close, and the thread will exit.

5. **Main:** This module implements the main execution routine and brings all other modules together. It monitors a given folder for file creation events using Linux inotify API. In case of an event, a *Transmission* thread is created and assigned to the newly created file, then network's parameters and flow's information are fetched through the *Network probing* and *Policies* modules, respectively. The NIS is executed once the previous information is available, and the results are fed to the Transmission thread to start sending the data. It also routinely monitors network events such as newly discovered networks or network disconnection, and reacts by forcing the network probing execution followed by NIS, the new results are propagated to the active *Transmission* threads.

Table 6: QoS, monetary and power consumption values of the used networks

Technology	Carrier	Dynamic*				Static	
		Throughput (Mbits/s)	Latency (mSec)	Jitter (mSec)	PER (%)	Cost (cents/Mbyte)	Power consumption (mWatt)
4G	Bouygues	12	83	10	0	0.2	1726
3G	SFR	3.5	345	17	0	1	1678
Wi-Fi	Lab WiFi	5	65	23	1	0.001	210

* *Values are for reference only, since they are subject to change and need to be computed during the program execution.*

7.2. Server

The server uses Linux Ubuntu 16.04.4. The implementation is fairly simple, it accepts connection from clients and stores the received files locally. If a file already exists the server responds with the end-of-file position to the client. This is useful in case of disconnection so the client can resume sending from the last successfully sent packet. Another objective is to provide iperf3 server for network testing, it does so by running iperf3 server on multiple ports if possible for concurrent testing. The number of ports is limited by the server's connection uplink and/or downlink capacity, depending on the test direction. The connection's throughput is required to be at least equal to the clients' connections throughput combined for the measurement to be accurate. In this case study, the server is running on Amazon EC2 with 70 Mbits/sec downlink and 65 Mbits/sec uplink Internet connection, which is more than enough to provide concurrent testing for one client with three Internet connections rating at 20 Mbits/sec max.

The testbed's different modules and their interaction is represented through a component diagram in Figure 4.

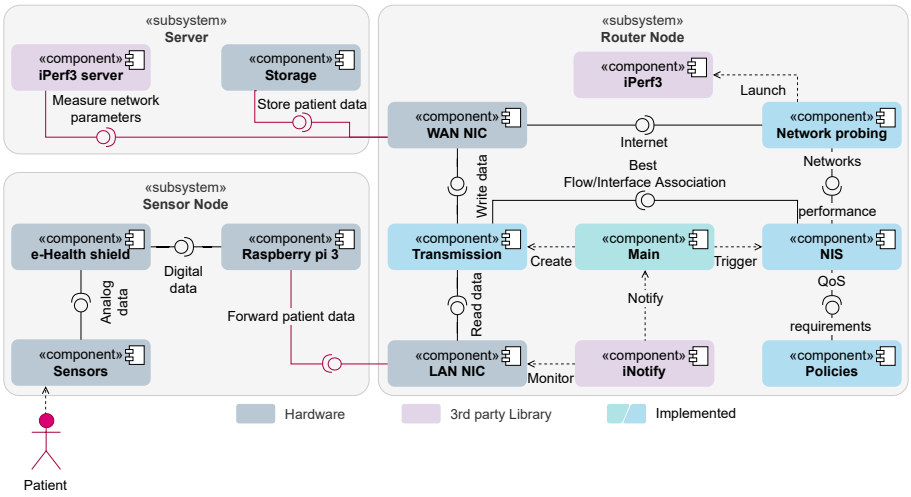

Figure 4: Testbed component diagram.

8. Results

In this section, we will discuss the testbed setup, execution and the obtained results. The real traffic through the Internet is used, with the clients located in Paris, France, and the server in Ohio, United States. The following process was followed in order to setup the testbed before each test case:

1. The captured data from each sensor is stored in a file, and the same file is used for all subsequent tests, in order to prevent data size changes between executions.

2. The router node is configured in an initial state where the sensors' files are already in local storage. Thus, isolating any variations introduced by wireless communications between the sensor and router nodes.

3. The router node is not connected to any wireless access network and no information is transferred.

4. The flow requirement is parameterized according to Table 7. While the user preference is set to ***Cost*** for all traffic.

5. The network interfaces are connected and the scenario is executed.

The execution starts with the router node initiating a network probing and collects all the information about the network's parameters, then running one of the NIS algorithms. The selection algorithms except Brute-force can run indefinitely, therefore a stopping condition is needed to terminate the algorithm's execution. We limited the running time to 20 seconds or when the results seize improving for 100 consecutive iterations.

Table 7: Sensors' network requirements

Sensor	File size (Mbyte)	Requirement
Airflow	1	Real-time data
Thermometer	4.5	Real-time data
EMG	3.5	Best effort data
GSR	2	Best effort data
ECG	55	Scavenger

Once an association is obtained, each file is sent through the assigned network. Results are acquired after all data is transferred successfully. Network disruptions during execution will affect the obtained results and, as such, are discarded.

Results contain five parameters:

1. Satisfaction ratio: represents to how extent the user's and application's (flows) requirements are met. 1.0 means all the requirements are satisfied, while 0.0 is the antithesis.
2. Sending time: measures the time spent successfully sending all the data.
3. Cost: is the data charges in €0.01.
4. Energy consumption by active network interfaces.
5. Selection time: represents the selection algorithm's running time in second.

For each selection algorithm, there are three different outputs, representing the number of sensor nodes used in the test, 1, 2 and 3 nodes which yield 5, 10 and 15 flows, respectively. Each test case is repeated ten times and results are averaged. Output is depicted in Figures 5-9.

Figure 5 shows the satisfaction ratio (SR). Using 5 flows, all selection algorithms produce SR equal to 1, except the random solution. By increasing the number of flows, the SR starts to drop reaching its lowest value 0.67 with the random solution while TABU and SA perform slightly better at 0.75 and 0.71, respectively. The STS gives results identical to the Brute-force algorithm.

Sending time is affected by the FIA i.e., assigning flows with high bandwidth demand to networks with large throughput takes less time to send the data than doing otherwise. In this testbed, the ECG file is bulky and the flow type is set as the scavenger, which prioritizes the network throughput. Therefore, it makes sense to assign it to the network with the largest throughput (4G) while still considering other flow requirements. With small problem size, as in the case of 5 flows, the objective is easy to attain, and all five algorithms give close results as shown in Figure 6. But as the problem size grows larger, the resources become scarce and the goal is increasingly difficult to reach, while considering the flow requirements and user preferences. Both Brute-force and STS give the shortest

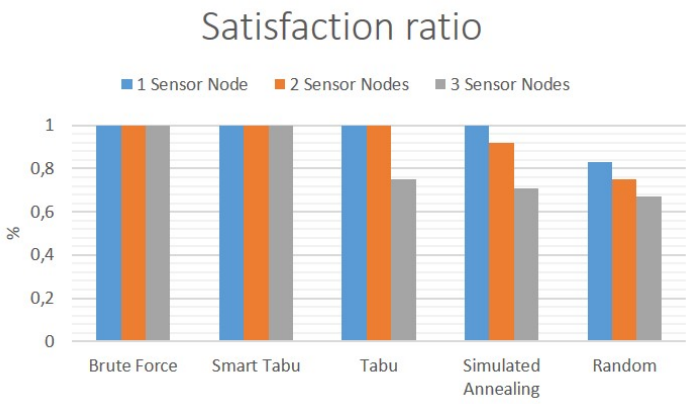

Figure 5: Satisfaction ratio using different selection algorithms while increasing sensor nodes.

time, while the other algorithms start to suffer gradually as the problem size increases, with random strategy giving the worst results.

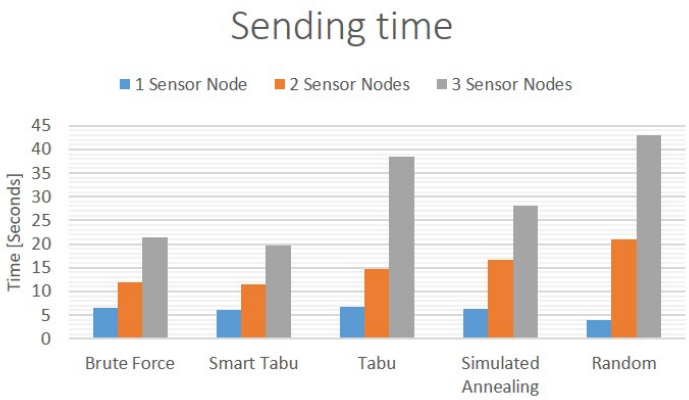

Figure 6: Time to send all files using different selection algorithms while increasing sensor nodes.

Cost is inversely proportional to the sending time, and that is due to the type of networks and user preference used. The network with the highest throughput is the 4G network, but has slightly elevated data charges, while Wi-Fi has the lowest price at an average throughput. Therefore, since user preference is set to Cost, all selection algorithms except Brute-force and Random, will lean at first toward solutions with low cost, even at the expense of SR, then start converging toward the global optimum, which may require the usage of the 4G network or even 3G to obtain the highest SR at the lowest possible cost.

When only 5 flows are considered, reaching the global optimum is a quick

process, and all algorithms give results comparable to that of the Brute-force, except the Random solution. But as the search space grows larger, both TABU and SA produce results with lower cost than that of the Brute-force, but worst in terms of SR and sending time, which suggest that both algorithms are stopped before reaching (slow at finding) the global optimum. The STS gives the same results as the Brute-force in all three cases.

Energy consumption follows the same trends, since the 4G and 3G networks also have higher power consumption than Wi-Fi. But it should be noted that we considered the energy consumption of the network interfaces only. If we were to factor the computational energy expenditure, solution with high sending time actually has a higher energy consumption.

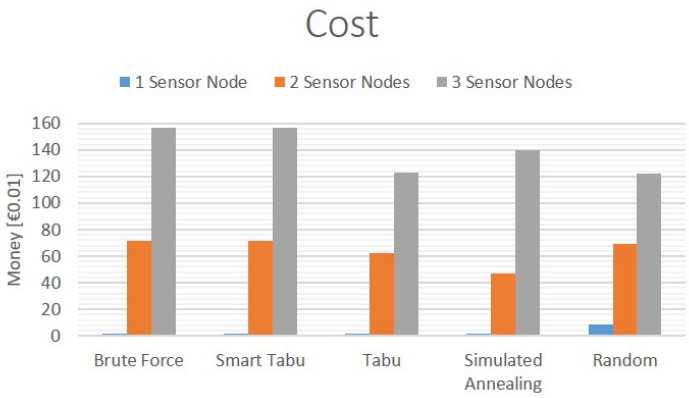

Figure 7: Charges using different selection algorithms while increasing sensor nodes.

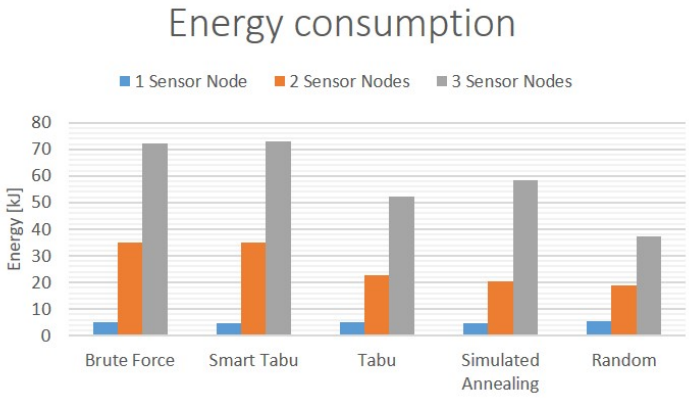

Figure 8: Energy expenditure using different selection algorithms while increasing sensor nodes.

While finding or approaching the global optimum is appealing, the time required to obtain such a result is an important factor to determine if the algorithm is suitable for real scenario usage. Random approach's execution time is negligible, since there is no search involved, but the solution is also of poor quality compared to other approaches. Brute-force always gives the global optimum, but the required time to obtain the result grows exponentially when the problem size increases as can be seen in Figure 9. When 5 flows are considered, the running time is equal to 244 ms and grows to 57 s and 14 hours (omitted from the graph since it dwarfs other values) for 10 and 15 flows, respectively, which render the algorithm completely impractical especially when the terminal is mobile.TABU and SA take less time, but the quality results start deteriorating as the problem size grows. The STS provides best of both, by producing comparable results to the Brute-force at a fraction of the time, 7.43 s when considering 15 flows, which is half the time taken by the TABU.

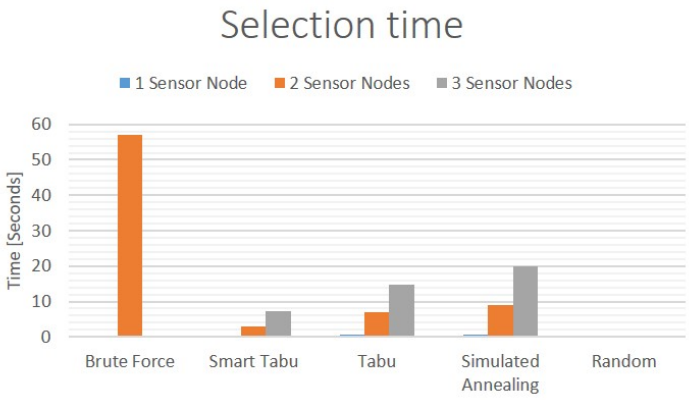

Figure 9: Different selection algorithms running time while increasing problem size.

9. Conclusion and future work

In this paper, we work towards solving the FIA problem in HWNs. Different random search approaches, such as TS and SA, in the context of FIA, were studied in a simulation-based comparative manner. The simulation outcomes shed light on weak and strong points of two approaches in the study scope context and provoke our proposed approach that aims to alleviate the deficiencies. We proposed three improvements to the original TS to provide an efficient FIA. The effectiveness and efficiency of the proposed approach were firstly evaluated through simulation and the obtained simulation results were supported by comparison with the original TS, SA and an alternative existing work in the literature. The obtained promising results indicate that the proposed approach far outperforms the other approaches. In order to further quantify the actual benefit of the proposed approach, we have developed a real health monitoring

testbed, comprising multiple sensors for medical application, which can be used for healthcare facilities due to several factors such as, emergency, remote location, limited mobility, being part of the daily routine (patients with chronic conditions), or simply because it is a tedious expensive task for some simple procedures. The obtained results were reported, analyzed and compared with other existing works. The results of these further experimental tests confirm the simulation results.

The future work can focus on further investigation of the proposed algorithm's performance. Instead of adhering to the assumption made here, that a flow can be assigned to only one network interface, a terminal with link aggregation capability can be considered where multiple network connections are combined in order to increase the throughput beyond what a single connection could sustain, and to provide enough redundancy in case one of the links should fail. An in-depth investigation can then be carried out, to determine the impact on energy consumption, monetary charges, performance, and user satisfaction.

[1] V. Gazis, N. Alonistioti, L. Merakos, Toward a generic 'always best connected' capability in integrated wlan/umts cellular mobile networks (and beyond), IEEE Wireless Communications 12 (3) (2005) 20–29.

[2] M. A. Senouci, M. S. Mushtaq, S. Hoceini, A. Mellouk, Topsis-based dynamic approach for mobile network interface selection, Computer Networks 107 (2016) 304–314.

[3] F. H. Mirani, N. Boukhatem, P. N. Tran, On terminal utility for multiple flow/interface association in mobile terminals, in: IFIP Wireless Days (WD), Niagara Falls, Ontario, Canada, 2011, pp. 1–6.

[4] O. Ormond, J. Murphy, G. m. Muntean, Utility-based intelligent network selection in beyond 3g systems, in: Proceedings of IEEE International Conference on Communications, Vol. 4, Istanbul, Turkey, 2006, pp. 1831–1836.

[5] K. Ahuja, B. Singh, R. Khanna, Particle swarm optimization based network selection in heterogeneous wireless environment, Optik - International Journal for Light and Electron Optics 125 (1) (2014) 214–219.

[6] J. Mcnair, F. Zhu, Vertical handoffs in fourth-generation multinetwork environments, IEEE Wireless Communications 11 (3) (2004) 8–15.

[7] H. Yu, B. Zhang, A heterogeneous network selection algorithm based on network attribute and user preference, Ad Hoc Networks 72 (2018) 68–80.

[8] H. Yu, Y. Ma, J. Yu, Network selection algorithm for multiservice multimode terminals in heterogeneous wireless networks, IEEE Access 7 (2019) 46240–46260.

[9] M. A. Senouci, S. Hoceini, A. Mellouk, Utility function-based topsis for network interface selection in heterogeneous wireless networks, in: Proceedings of IEEE ICC, Kuala Lumpur, Malaysia, 2016, pp. 1–6.

[10] A. Roy, P. Chaporkar, A. Karandikar, Optimal radio access technology selection algorithm for lte-wifi network, IEEE Transactions on Vehicular Technology 67 (7) (2018) 6446–6460.

[11] E. Stevens-Navarro, Y. Lin, V. S. Wong, An mdp-based vertical handoff decision algorithm for heterogeneous wireless networks, IEEE Transactions on Vehicular Technology 57 (2) (2008) 1243–1254.

[12] Z. Ning, Q. Song, Y. Liu, F. Wang, X. Wu, Markov-based vertical handoff decision algorithms in heterogeneous wireless networks, Computers & Electrical Engineering 40 (2) (2014) 456–472.

[13] L. Chen, H. Li, An mdp-based vertical handoff decision algorithm for heterogeneous wireless networks, in: Proceedings of IEEE Wireless Communications and Networking Conference, Doha, Qatar, 2016, pp. 1–6.

[14] J. Hou, D. O'Brien, Vertical handover decision making algorithm using fuzzy logic for the integrated radio-and-ow system, IEEE Transactions on Wireless Communications 5 (1) (2006) 176–185.

[15] R. K. Goyal, S. Kaushal, A. Sangaiah, The utility based non-linear fuzzy ahp optimization model for network selection in heterogeneous wireless networks, Applied Soft Computing 67 (2018) 800–811.

[16] E. Watanabe, D. S. Menasche, E. de Souza e Silva, R. M. M. Leao, Modeling resource sharing dynamics of voip users over a wlan using a game-theoretic approach, in: Proceedings of 27th IEEE Conference on Computer Communications (INFOCOM), Phoenix, AZ, USA, 2008, pp. 915–923.

[17] M. Cesana, N. Gatti, I. Malanchini, Game theoretic analysis of wireless access network selection: Models, inefficiency bounds, and algorithms, in: Proceedings of the 3rd International Conference on Performance Evaluation Methodologies and Tools, Athens, Greece, 2008, pp. 1–10.

[18] M. A. Senouci, M. R. Senouci, S. Hoceini, A. Mellouk, An evidential approach for network interface selection in heterogeneous wireless networks, in: Proceedings of IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 2016, pp. 1–7.

[19] E. van den Berg, P. Gopalakrishnan, B. Kim, B. Lyles, W. I. Kim, Y. S. Shin, Y. J. Kim, Dynamic network selection using kernels, in: Proceedings of IEEE International Conference on Communications, Glasgow, UK, 2007, pp. 6049–6054.

[20] L. Ozbakir, A. Baykasoglu, P. Tapkan, Bees algorithm for generalized assignment problem, Applied Mathematics and Computation 215 (2010) 3782–3795.

[21] L. Breslau, S. Shenker, Best-effort versus reservations: A simple comparative analysis, in: ACM SIGCOMM, Vancouver, British Columbia, Canada, 1998, pp. 3–16.

[22] S. Shenker, Fundamental design issues for the future internet, IEEE Journal on Selected Areas in Communications 13 (1995) 1176–1188.

[23] Mysignals, https://www.cooking-hacks.com/mysignals-sw-ehealth-medical-biometric-complete-kit.

[24] Connection bridge, https://www.cooking-hacks.com/raspberry-pi-to-arduino-shield-connection-bridge.

[25] Iperf3, https://iperf.fr/, iperf3.

[26] Owamp, http://software.internet2.edu/owamp/.