Flow/Interface Association for Multihomed Mobile Terminals in Heterogeneous Wireless Networks

Mohamed Abdelkrim Senouci*, Hadj Senouci†, Said Hoceini*, and Abdelhamid Mellouk*
*Université Paris-Est, LISSI, UPEC, 94400 Vitry sur Seine, France
†Algérie Télécom, Saida, Algeria

***Abstract*—Wireless access networks (e.g., Wi-Fi, UMTS, and LTE) have different characteristics in terms of coverage area, cost, and different capacities to cater for diverse service needs. These networks may cover the same region forming a heterogeneous wireless environment. Mobile users run different kinds of service over the Internet (e.g., video streaming, interactive video gaming, etc.). These services lead to diverse flows with different requirements in terms of Quality of Service (QoS). Current terminals (e.g., laptops and cell phones) are increasingly being provided with multiple wireless access network interfaces (e.g., UMTS, LTE, Wi-Fi) for users to not be tied to any one access network, but instead be free to switch among available networks. In this multi-technology, multi-application and multi-user environment, it would be advantageous for multihomed terminals to simultaneously use multiple networks instead of switching from one interface to another. This involves the association of each flow with a network that meets its specific requirements in a way that best maximizes the global terminal utility. Associating each flow with a suitable interface is considered as a particular case of network interface selection and is called flow/interface association throughout this paper. The main target of the flow/interface association is to satisfy individual flow's requirements while maximizing the global system performance. The flow/interface association is an optimization problem, and more specifically is related to stochastic optimization problems. This paper proposes a new flow/interface association scheme for multihomed mobile terminals in heterogeneous environment. We provide analytical studies and simulation experiments to demonstrate the efficiency of the proposed approach.**

***Index Terms*—Heterogeneous Wireless Networks, Network Interface Selection, Flow/Interface Association, Stochastic Optimization.**

I. Introduction

The mobile terminals are different in capacities and can run simultaneously different types of application over Internet (e.g., interactive video gaming, streaming, VoIP, etc.) which leads to diverse flows with different exigencies in terms of quality of service (QoS) such as delay, jitter, packet loss rate, and throughput. These flows may need different kinds of communication technology. Therefore, it would be advantageous for the multihomed terminals to associate each data flow with a network that meets its specific requirements. In such an environment, the major issue is the flow/interface association, where the main target is to satisfy the individual flow's requirements while maximizing the global system performance; hence the association solution that provides the maximum total utility will be selected. Associating each flow with a suitable interface is considered as a particular case of network interface selection [1], [2], [3] and is called flow/interface association throughout this paper.

In an overlapping area, the multihomed mobile terminal has the possibility to use multiple interfaces simultaneously, where each particular application flow may be associated with its suitable network while maximizing the global terminal utility. In [4], a policy-based approach has been proposed that assigns application flows to different interfaces, where each interface is selected dynamically for a particular flow using the user-defined policies. This solution does not consider the tradeoff between metrics considered for network selection, where the relative importance of each criterion to the selection problem is not specified. Another approach to split flows over network interfaces is proposed in [5], where only the application metrics were considered rather than network metrics (QoS). Authors claim that the decisions based on network metrics may not lead to the best user perceived quality. In [6], a knapsack model based-scheme has been proposed for the flow/interface association. A knapsack problem is a combinatorial optimization problem that models a situation similar to filling a knapsack with all or part of a given set of objects each having a weight value, without exceeding the capacity of the knapsack. Objects placed in the knapsack must maximize the total value without exceeding the maximum weight. The purpose of mapping the flow/interface association problem to a knapsack problem is to allocate applications to networks. The objective is to maximize the total profit without surpassing the capacity of any of the knapsacks. The user utility is the profit produced by the items placed in the knapsacks. In flow/interface context, knapsacks are the networks, items are the flows of applications, and the capacity of the knapsack is the resource constraint of the network. In [7], a meta-heuristic approach based on Tabu search has been proposed. The authors deemed the standard diversification ineffective, and proposed a new one named "Oriented diversification" that improved the algorithm's performance.

In this paper, we tackle the flow/interface association problem as a stochastic optimization problem. More precisely, this paper makes the following specific contributions. First, we present a review of most and recent used methods for flow/interface association problem. Second, we discuss the flow/interface association concept and we present the model description. Third, we present and compare two random search methods widely applied to stochastic optimization problems. Fourth, we present our flow/interface association scheme. Finally, we present and discuss obtained results.

The rest of this paper is organized as follows: Section II presents the flow/interface association concept, while Section III provides a comparison between Tabu Search and Simulated Annealing algorithms in the context of flow/interface association. Section IV details our proposed approach. Experiment results are presented and discussed in Section V. Finally, Section VI concludes the paper.

II. Flow/Interface Association

A. Model description

Let us consider a multihomed mobile terminal located in a zone covered by several RATs, and running several different applications. Our goal is to associate each flow of application with its suitable available network in a way that maximizes the global terminal utility.

Assume that we have n available networks discovered by the mobile terminal, where $I = \{I_i, i = 1, 2, ..., n\}$, and m data flows of different applications run by terminal, where $F = \{F_i, i = 1, 2, ..., m\}$. The terminal can have a set of n^m possible association, where $A = \{A_i, i = 1, 2, ..., n^m\}$.

Each association solution $a_i \in A$ allows the set of flows m to connect to the set of interfaces n. such that each flow $f_j \in F$ is assigned to one and only one network $i_k \in I$ and each network i_k can accept 0 to m flows. For example, an association solution a_i allows flow f_1 to connect to interface i_1, flow f_2 to interface i_3, flow f_3 to interface i_3, ..., and flow f_m to interface i_n.

Let $Q_{k,j}$ (respectively $C_{k,j}$ and $E_{k,j}$) denote the QoS (respectively cost and energy consumption) utilities of assigning flow f_j to a network i_k determined by the association a_i. the utility for flow f_j using the network i_k is described as follows:

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

Where α, β and γ are the weights that indicate the relative importance between the QoS, cost and the energy consumption respectively.

A utility value U_i is calculated for each association solution a_i and is denoted by $U_i(a_i)$. For an association solution a_i, the overall terminal utility is computed as follows:

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

The objective is to maximize the global terminal utility; hence select the association solution that provides the maximum total utility.

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

The interface utility function describes the satisfaction level of the flow(s) associated with a network interface. It is worth mentioning that an association solution with the largest global utility can be chosen as the best solution only if it is valid association, which means all interfaces must fulfill all criteria requirements of the flows assigned to them. Otherwise, the whole association will be considered invalid (this part is further explained in Section III-A).

In the sequel, we present two random search methods widely applied to stochastic optimization problems.

B. Random search algorithms

The random search algorithms are more complete and complex than simple heuristics. They usually provide a solution of very good quality for problems arising from the fields of operational research or engineering. Problems for which there are no simple effective methods for solution, or when the resolution of the problem requires time or large storage memory.

The relation between the execution time and the quality of the solution found by a meta-heuristic remains in most cases very interesting compared to different types of resolution approach.

The methods that iteratively try to improve a solution are called local search methods or trajectory methods. These methods construct a trajectory in the space of solutions by attempting to move towards the optimal solutions. The best known examples of these methods are: Simulated Annealing and Tabu Search. The key concepts of each algorithm are presented in the following.

1) Simulated Annealing: The principle of simulated annealing is to traverse the solution space iteratively. We start with a solution denoted s_0 initially generated in a random manner, of which corresponds an initial energy E_0 and a generally high initial temperature T_0. At each iteration of the algorithm, an elementary change is made on the solution. This modification varies the energy of the system ΔE. If this variation is negative (the new solution improves the objective function, and reduces the energy of the system), it is accepted. If the solution found is worse than the previous one, then it will be accepted with a probability P calculated according to the following Boltzmann distribution:

$$P(E, T) = exp^{\frac{-\Delta T}{T}}$$

This idea of an annealing process of a melt is used and applied in stochastic problems to obtain an optimal result [8].

2) Tabu Search: This method is a local search method combined with a set of techniques to avoid being trapped in a local minimum or repeating a cycle. This method has shown great efficiency in solving difficult optimization problems. Indeed, from an initial solution s in a set of local solutions S, subsets of solution $N(s)$ belonging to the neighborhood S are generated. Through the evaluation function we retain the solution which improves the value of f chosen from the set of neighboring solutions $N(s)$.

The algorithm sometimes accepts solutions that do not always improve the current solution. The Tabu list T of length k contains the last k visited solutions, which prevents k previously explored solutions to be accepted and stored in the Tabu list. Then the choice of the next solution is made on a set of neighboring solutions outside the elements of this Tabu list. When the number k is reached, each selected new solution replaces the oldest one in the list. The construction of the Tabu list is based on the FIFO principle. As a stop criterion, it is possible, for example, to set a maximum number of iterations without improvement of s^*, or to fix a time limit after which the search must stop.

III. Algorithms comparison

Two algorithms are considered in this study, namely Tabu Search from the stochastic class and Simulated Annealing from the physical class. The enumeration algorithm is used to obtain the global optimum, and also as reference against which other algorithms are assessed. The implementation and simulations are achieved with pure python (python 2.7). The simulation outcomes will shed light on the two approaches' strengths and weaknesses in the study scope context and motivate our proposal that aims to remedy the shortcomings.

A. Preliminaries

For the data flows, we consider seven types distributed among three classes. Class 1 is "Critical" which comprises PCM VoIP (G.711) and Telepresence, class 2 "Greedy" includes FTP, P2P, Downloading, and class 3 "Elastic" covers video and audio streaming. Every class will receive a different QoS settings, class 1 will use Real Time, class 2 Scavenger, and class 3 Best Effort, and each type will define the requirements of 4 criteria (Bandwidth [Mbps], Latency [ms], Jitter [ms], Loss [PER]) by which the interface utility is measured.

The terminal is equipped with I interfaces ($I \geq 2$) and can have multiple interfaces with the same radio access technology, each interface can accommodate F data flow(s) ($F \geq 0$). Every interface will provide data about six criteria (Bandwidth [Mbps], Latency [ms], Jitter [ms], Loss [PER], Cost [cent/MB], and Energy [mW/Mbit]). Six radio technologies are defined including WiFi 802.11b, 802.11a, 802.11n, GSM, UMTS, and LTE. For each radio technology the criteria values are chosen from a predefined interval, so as to make it easier to generate same radio but with different specs.

The user preferences are determined with three values (weights) for three variables, QoS, cost, and energy-consumption, the sum of these values equals to one, and the value of a variable is directly proportional to its importance. The QoS represents the weight of flow or application requirements while cost (resp. energy consumption) represents the weight of user (resp. terminal) requirements.

Interfaces that are assigned to more than one application will allocate resources according to the applications combined requirements. If the interface fails to fulfill one or more criteria requirements, then the whole association is considered invalid. The invalid association can be managed in two ways.

a) The algorithm keeps track of both valid an invalid association, since some invalid associations will have higher utility than valid ones, the utility value may not be suitable for comparison between valid and invalid solutions. This approach will offer the user with two choices, so he/she might choose an invalid association if it has a very high utility, but it has slightly higher CPU and memory footprint since two solutions are maintained.
b) Compute the number of violations for an interface and use it to normalize the utility, so that valid association will always have higher value than invalid one. With this method the utility can be used to find the best association, but it robs the user from deciding.

B. Simulation scenarios

In the first scenario, we consider six applications (2x VoIP, Telepresence, P2P, video streaming, and audio streaming) and a terminal equipped with four interfaces (11g, 11n, 11ac, and LTE). We have a total of 4096 possible associations, we refer to each one as a node. The algorithms will explore a preset number of nodes, during which each one will attempt to find the global optimum. The algorithms performances are compared against enumeration algorithm and measured by three criteria (see Table I).

- Global optimum probability $P\%$ [0, 1]: it measures the percentage of an algorithm to find the global optimum.
- Utility deficit UD [0, 1]: for solutions that are not global optimum it makes sense to measure how far they are from it. Value of 1 means the algorithm found no solution and the lower the value gets the better the solution is, while the value of 0 means the solution is a global optimum.
- Speedup SP: represents the algorithm speedup gain compared to enumeration.

In the second scenario, we increase the number of interfaces to five (11g, 11n, 11ac, LTE, and UMTS) and flows to ten (2x VoIP, 2x Telepresence, 2x P2P, 2x video streaming, and 2x audio streaming) which amount to 9765625 possible associations.

All results shown hereinafter (unless pointed out) are an average of executing the algorithms N times, each time using the same set of interfaces but with different specs.

N equals 100 for the first scenario and 25 for the second. The average running time of both scenarios using enumeration algorithm is 0.5326 seconds and 1776.8030 seconds, respectively.

NB. Setting the number of allowed nodes to be explored as a stopping criterion is more accurate than the number of iterations, since different algorithm will explore different number of possibilities during each iteration.

C. Simulation results

TABLE I: Scenario 1-results for Tabu Search & Simulated Annealing

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

From Table I, it is clear that increasing the number of explored nodes enhances $P\%$ and UD while affects negatively speedup, which is to be expected since exploring more nodes means more running time and better chance to improve solutions and find optimum. Second observation is that Tabu search (TS) performs better than Simulated annealing (SA). But the overall observation is that even by exploring about 50% of all possibilities both algorithms offer low quality solutions with

very modest speedup. To improve their performance, we make two minor modifications (results are shown in Table II):

- Neighborhood by displacement: instead of generating the neighborhood randomly, we take the current solution and assign each flow to all available interfaces one at time. For illustration, we consider two interfaces and three flows, and current solution = [1, 1, 2]. The solution means flows 1 and 2 are assigned to interface 1 and flow 3 is assigned to interface 2. Neighborhood = [[2, 1, 2] [1, 2, 2] [1, 1, 1]].
- Caching: both algorithms (SA more than TS) will revisit some solutions hence their neighborhood, if we cache some of the neighborhood results, that should improve the algorithms speedup.

TABLE II: 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 improvement in $P\%$ and UD is obvious with SA reaching its peak far more quickly than TS at around 200 nodes then stuck on local optima, which is a typical behavior for SA as the temperature parameter gets lower, the algorithm tends to seize exploration and stick to the current best solution. For TS, although the improvement is slower, the algorithm manages to find all global optimum. These results prove two points:

- The way the neighborhood is generated impacts the outcome heavily.
- 100% success rate is costly

As for the caching, it mostly benefits SA and that is due to nature of the algorithm, since it lacks a mechanism to prevent looping over the same search space, revisiting previously explored nodes is probable. For TS, the caching did not add much since the algorithm prevent revisiting nodes by maintaining the Tabu list. The slight improvement is due to revisiting nodes that are generated from solutions that are not Tabu and/or the Tabu list is updated and the solution previously visited is no longer considered Tabu. Since the advantage of making these minor modification is apparent, and in order to provide fair results; all subsequent algorithms against which we will compare our proposal will adopt these modifications, including Tabu with oriented diversification [7].

The performance of both algorithms improved, when solving small scale problems, but when handling large problem as in the second scenario, the performance tends to decline (see Table III). The $P\%$ is directly proportional to the number of nodes explored, but the larger the set of nodes gets the more time it takes, which is unproductive in the context of flow/interface association, since the user will strive to obtain the maximum utility but only as long as it is achievable during a certain time frame. Another factor is the dynamic nature of mobile networks. It requires real time results to make use of them, as it is futile to use the result while some networks are no longer available and/or other networks are within reach.

TABLE III: 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

IV. Our proposal

In this section, we seek to remedy the shortcomings indicated previously. We chose Tabu search as the main algorithm, although it has lower performance compared to Simulated annealing, 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 three improvements to the original Tabu search as follows.

A. Clever Diversification

TS uses random diversification to escape being stuck on local optima, 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 lower rate. This will increase the chance to stumble upon a neighborhood with 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) Limit the search space, using the pseudo-code as defined in Algorithm 1. The algorithm's output is a matrix M, where each column represents a flow and the lines are the available interfaces for that flow.

Algorithm 1 Limit Diversification Space

Require: M empty matrix
for f in Flows **do**
  for i in Interfaces **do**
    Compute the utility U_{fi} using equations [2]:
    for upward parameters (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 parameters (e.g., Jitter)

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

    for monetary cost:

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

  end for
  Compute 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

e.g.

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

means flow 1 can use only interface 1, and flow 2 can choose interface 1, 2 or 3 and so on.

2) Generate a candidate: for each flow (f) we randomly assign an interface (i) as defined in M. If the resulted candidate is tabu, we repeat the process for a maximum of n times ($n > 0$). If no candidate can be generated then resort to random diversification.

B. Approximate Initial Solution

Tabu search generates a random candidate for an initial solution and starts improving upon it, sometimes the provided solution is too far from the global optimum and/or invalid and the algorithm will spend a considerable amount of time to improve it. Instead, we supply the algorithm an approximate solution with high utility and valid (if available), so the algorithm has a better chance to find global optimum.

To obtain an approximate solution for each flow, we locate the interface with the highest utility and assign it to that flow, then update the interface's consumable resource (e.g., bandwidth, buffer, etc.) to reflect the allocating. The whole process will require $I * F$ evaluations. Although it is a simple technique, it does improve the performance dramatically as we will demonstrate it in the next section.

C. Oscillation

By default, TS only accepts solutions that are equal or have better utility, this behavior renders the algorithm greedy and it might get stuck on local optima. This is where diversification might help by providing random candidates that might lead to the global optimum, but this is only true if the new candidates or their neighborhood contain solutions equal or better than the current best, when there is no improvement the diversification is repeated. This loop gets longer the closer the current solution is to the global optimum, that's because the number of candidates better than the current solution gets lower.

To clarify, let us consider an example. Assume the current solution is second to global optimum, and in order to reach the optimum, 2 flows must change their interfaces. Generating the neighborhood by displacement (as we do) will only change one flow at time, which means the algorithm will never reach the global optimum, since it will not explore solutions with lower utility (see Table V).

This problem can be easily solved by allowing TS to accept and explore sub-optimal solutions. To achieve that, we use a mechanism similar to SA's acceptance probability (Equation 6), but unlike SA the probability here increases when the number of iterations since last improvement gets larger (resp. c is set low). This will result in TS to be more stochastic when improvements ceased. Once a better solution is found, then the probability is reset to allow the algorithm to explore the vicinity of the current solution.

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

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

V. Performance evaluation

We incorporate the improvements discussed in the previous section into TS, the resulted algorithm will be referred to as Clever Tabu search (CTS). The proposed algorithm is compared against TS with oriented diversification (OTS) [7] running the same scenarios presented above. Results in Table IV and Table V highlight the performance gain of the proposed TS.

TABLE IV: OTS *vs.* CTS – Scenario 1

N (Nodes)	Oriented Tabu			Clever Tabu		
	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

Looking at the results, it is apparent that OTS does not offer much of an advantage over standard Tabu, and that is due to the way oriented diversification works. It takes the $TabuList$ as an input and looks for flows that have not visited all interfaces, and randomly assigns one of the unvisited interfaces to one of the flows, if none can be found then it uses random diversification. The approach is of little help when the candidate's neighborhood is generated through displacement, since most of the oriented diversification results are already evaluated when exploring the neighborhood. We tried generating the neighborhood randomly, but the algorithm performed even poorly compared to TS. However, there is one particular case where this approach can be helpful, and that's when the algorithm meets a similar situation as the example discussed on the previous section under Oscillation, which is the case in Table V, where OTS found all global optimum by evaluating 50k nodes as opposed to TS where the search halts at 96% nonetheless TS will still converge albeit slowly.

As for CTS the results are overwhelming, in the first scenario the algorithm starts at 99% success rate, thanks to the approximate initial solution, and reaching 100% after evaluating 100 nodes only. This proves that providing good initial solution is fundamental. Another observation is that CTS has slightly lower speedup compared to OTS and that is expectable, since the algorithm spends a considerable time to compute the initial solution and limit the search space for the diversification; but that is visible only on small problems, as the search space becomes larger the overhead is negligible. In the second scenario, the first case demonstrates that CTS can solve large problems quickly without hindering its performance, but although the algorithm converges quickly, sometimes it does get stuck on local optimum early on and cannot escape no matter how long it takes. The second case depicts the importance of oscillation, by simply allowing the search to accept and explore sub-optimal solutions, the algorithm can avoid being stuck on local optima.

TABLE V: OTS *vs.* CTS – Scenario 2

Nodes	Oriented Tabu			Clever Tabu					
				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			

VI. Conclusion

In this paper, a new efficient flow/interface association approach was proposed to allow the multihomed mobile terminal to associate each data flow with its suitable network interface in a way that best satisfies all the flows and maximizes the global terminal utility. A comparative study of the random search approaches including Simulated Annealing and Tabu Search in the context of flow/interface association was provided. Moreover the proposed approach was compared to an existing work in the literature. Simulation results show the effectiveness and efficiency of the proposed approach and that the latter far outperforms the other approaches. Also the results of the proposition are obtained within short time frame (sub 10 ms even with the inherit slowness of Python) which makes it fit for real time applications.

References

[1] M. A. Senouci, M. S. Mushtaq, S. Hoceini, and A. Mellouk. Topsis-based dynamic approach for mobile network interface selection. *Computer Networks*, 2016.

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

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

[4] J. Ylitalo, T. Jokikyyny, T. Kauppinen, A. J. Tuominen, and J. Laine. Dynamic network interface selection in multihomed mobile hosts. In *Proceedings of the 36th Hawaii International Conference on System Sciences*, page 1–6, 2003.

[5] O. Mehani, R. Boreli, M. Malier, and T. Ernst. User- and application-centric multihomed flow management. In *2011 IEEE 36th Conference on Local Computer Networks*, pages 26–34, 2011.

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

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

[8] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *SCIENCE*, 220(4598):671–680, 1983.