Channel Assignment Problem in Cellular Networks: A Reactive Tabu Search Approach

Didem Gözüpek, Gaye Genç, and Cem Ersoy

Computer Networks Research Laboratory (NETLAB)
Department of Computer Engineering, Boğaziçi University, TURKEY
{didem.gozupek, gaye.genc, ersoy}@boun.edu.tr

***Abstract*—The channel assignment problem (CAP) in cellular networks is concerned with the allocation and reuse of the frequency spectrum to the base stations in such a way that both the interference constraints and the traffic requirements of the cells are met. In this paper, we apply a *reactive tabu search* based method to solve the CAP and compare this approach with the classical tabu search as well as genetic algorithm based approaches in the literature.**[1]

***Index Terms*—Channel assignment problem, frequency assignment problem, tabu search, reactive tabu search.**

I. INTRODUCTION

The development of the digital cellular phone standard GSM (General System for Mobile Communication) led to a rapidly increasing interest for frequency assignment. The fact that the governments have charged the operators for the usage of each single frequency separately introduced the need for the cellular operators to develop frequency plans that not only avoided high interference levels, but also minimized the licensing costs [1]. Consequently, the channel assignment problem (CAP), also referred to as the Frequency Assignment Problem (FAP), appeared as a major research problem that deals with the assignment of the frequency bands to the base stations. Although the CAP is a well studied optimization problem, engineers and researchers continue to show interest in the solution of CAP due to the ever increasing popularity of cellular communication systems.

Fixed Channel Assignment (FCA) deals with static models where the set of connections remains stable over time. On the contrary, Dynamic Channel Assignment (DCA) copes with the problem where the demand for frequencies varies over time. Hybrid Channel Assignment (HCA) combines FCA and DCA: A number of frequencies have to be assigned beforehand; however, a portion of the spectrum has to be reserved for the online assignment of frequencies upon request. In this work, we focus on the FCA problem.

The fixed optimal channel assignment in cellular networks is an NP-complete optimization problem with constraints. A multitude of approaches ranging from integer programming to neural networks and genetic algorithms have been proposed to address this problem [1]. In this work, we apply a reactive tabu search method [2]. We compare the performance of our approach with the classical tabu search based method in [3]. To the best of our knowledge, ours is the first work that concentrates on a reactive tabu search approach for the fixed optimal channel assignment problem.

The remainder of the paper is organized as follows: Section II provides the related work and Section III describes the problem formulation. Section IV introduces the classical and reactive tabu search approaches, and Section V presents the experimental results. Finally, Section VI concludes the paper.

II. RELATED WORK

Tabu search is a meta-heuristic technique that guides a local heuristic search procedure by local memory structures. The basic idea is to prohibit certain moves that would return to recently visited solutions, by rendering them tabu. The authors in [3] implement the tabu search algorithm to solve the CAP. The presented simulation results on benchmark CAPs reveal that their approach outperforms the existing methods.

In [4], a reactive tabu search method is employed for the general graph coloring problem, using the help of a partial solution. The authors state that the general approach is to start from an infeasible solution, and try to reach a feasible solution by changing the map coloring. Their solution considers partial but feasible solutions and tries to increase the size of the current partial solution.

In [5], the work which we adapt our problem formulation from, the authors use a simple genetic algorithm and a hybrid genetic algorithm for solving the CAP. They state that the simple genetic algorithm has convergence problems in large combinatorial problems. Therefore, the authors suggest the use of a hybridized genetic algorithm with integer coding using a local search algorithm as the mutation operator.

Integer programming is another method used for addressing the CAP in the literature. The authors in [6] start with a nonlinear problem, and linearize the objective function and the constraints by introducing new integer variables. Therefore they formulate the CAP as an integer linear programming problem. The constraints of this formulation are extracted from clique-like subproblems. This way, they determine the lower bounds for the sum of weighted constraint violations.

The authors in [7] employ an artificial neural networks based technique to solve the CAP. Their local updating rule, which is a function of the neighbor states and the coupling weights used to update the state of a neuron, consists of two terms. The first term is proportional to the demand deficiency, whereas the second term is is proportional to the distance violations.

[1]This work is supported by the State Planning Organization of Turkey under grant numbers DPT-03K 120250 and DPT-2007K 120610.

978-1-4244-5023-7/09/$25.00 ©2009 IEEE

ISCIS September 14-16, 2009
METU
Northern Cyprus Campus

They also propose several heuristics to increase the ability of escaping from local minima.

III. FORMULATION OF THE CHANNEL ASSIGNMENT PROBLEM

We have adopted the problem formulation described in (1) to (5) from [5] since it presents a simple yet satisfactory framework for the CAP. Suppose that the whole network is partitioned into N hexagonal cells each having one base station at the center transmitting with an omnidirectional antenna capable of tuning to any of the M available channels labeled as $m_k(k = 1, 2, ..., M)$. The interference between any pair of cells is assumed to be known so that the frequency separation constraints may be estimated in order to avoid co-channel and adjacent channel interference. These interference constraints are represented by a symmetrical interference matrix $\mathbf{X}$ shown in (1):

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ & & \ddots & \\ \vdots & \vdots & x_{ij} & \vdots \\ & & & \ddots \\ x_{N1} & x_{N2} & \cdots & x_{NN} \end{bmatrix} \quad (1)$$

where elements $x_{ij}(i, j = 1, ..., N)$ represent the frequency separation required between channels assigned to cells i and j necessary to maintain interference below a certain threshold. Using this matrix, it is possible to represent co-channel and adjacent-channel interference constraints by choosing proper values for the x_{ij} entries. In this work, we consider the co-channel interference case where the elements in X are given by:

$$x_{ij} = \begin{cases} 1; & \textit{if cell i and cell j cannot reuse a channel} \\ 0; & \textit{otherwise} \end{cases} \quad (2)$$

We refer to the requirements in (2) as *hard interference requirements*, since they are considered to be inviolable. Moreover, it is necessary to know the number of channels required in each cell in order to perform a channel assignment. If we denote the call arrival rate at cell i by λ_i and the mean call holding time for all calls by μ, then the Erlang-B formula enables us to determine the number t_i of channels demanded at cell i necessary to proportionate a grade of service equal to the probability of blocking P_b. Let $\mathbf{T}$ be a channel demand vector with elements $t_i(i = 1, 2, .., N)$ representing the number of required channels at cell i. The channel assignment problem is then defined as: Given M channels and N cells each requiring t_i channels, find the optimal $N \times M$ channel assignment matrix $\mathbf{A}$ given by:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ & & \vdots & \\ \vdots & \vdots & a_{ik} & \vdots \\ & & \vdots & \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{bmatrix} \quad (3)$$

where elements a_{ik} are as follows:

$$a_{ik} = \begin{cases} 1; & \textit{if cell i is assigned to channel } m_k \\ 0; & \textit{otherwise} \end{cases} \quad (4)$$

A channel assignment is admissible if both traffic and interference constraints are fulfilled. This implies that:

Traffic Constraint: $\sum_{k=1}^{M} a_{ik} = t_i, \forall i$

Interference Constraint: $|m_k - m_l| \geq x_{ij}$

where m_k and m_l are two channels assigned to cells i and j.

Consequently, the objective function that minimizes the number of constraint violations can be formulated as follows [5]:

$$F = \sum_{i=1}^{N} (t_i - \sum_{k=1}^{M} a_{ik})^2 + \sum_{i=1}^{N} \sum_{k=1}^{M} \sum_{\substack{j=1 \\ i \neq j}}^{N} \sum_{l=1}^{M} x_{ij} a_{ik} a_{jl} \quad (5)$$

where the first term is related to the violation of traffic requirements and the second term is related to the violation of interference constraints. In our implementation, we construct the initial solution by giving each cell the exact number of channels it requires and determine which channels to assign to a particular cell in a random fashion. In the subsequent iterations, we change the assignments in a way that preserves the traffic requirements of each cell. Therefore, the traffic requirement computation of the objective function becomes obsolete in our implementation. Hence, in the rest of the paper, we refer to the violation of interference constraints as the objective function.

IV. CLASSICAL AND REACTIVE TABU SEARCH

The classical tabu search is a meta-heuristic technique that guides a local heuristic search procedure to explore the solution space beyond a local optimality, by allowing a worsening solution [3]. The tabu list consists of a *short term memory* and a *long term memory* (LTM). The former refers to the moves that have been recently carried out, and the latter refers to the moves that have occurred too frequently. The results presented in [3] suggest that the use of a LTM does not improve the solutions much. However, we also implement the LTM to observe its impact on the performance.

A *move* is a small perturbation to the current solution. The *tabu tenure* specifies the number of iterations for which a move is considered to be a tabu, which can be roughly described as the length of the short term memory. The tabu search is stopped when one of the termination criteria is met; such as finding an optimal solution, time expiration, maximum number of iterations.

A. Classical Tabu Search

The initial solution is created by assigning random channels to the cells such that the channel demand of each cell is satisfied. In the CAP framework, a *move* is the exchange of a randomly selected unused channel with an in-use channel in a cell. Therefore, the set of possible moves in a cell is the exchange of the randomly selected unused channel with all in-use channels for that cell. In order to improve the solution,

a *neighborhood* is created by considering all possible moves for all cells. This means that the total number of moves in a neighborhood is equal to the total channel demand.

In order to choose the solution that yields the best improvement, the objective function is calculated with all the candidate solutions in the neighborhood. This is a costly operation; therefore, instead of calculating the objective function for all candidates, only the effect of the exchange is calculated, telling whether the solution has improved or not. The candidate solution that yields the best improvement in the objective function is accepted as the new solution. The result of the exchange is marked as *tabu*, so that this channel will not be considered for the neighborhood creation in the next iterations.

The number of iterations for which a move will be considered as a tabu is determined by the tabu tenure for each cell. With every iteration, the tabu value of an assignment decreases until it reaches zero. There are many approaches for determining the tabu tenure for each cell. If the tenure is very small, then a tabu move will quickly lose its tabu state and make the algorithm ineffective. If the tenure is very large, more and more channels will be marked as tabu and very few options will be left for neighborhood creation. The authors in [3] suggest (6) for computing the tabu tenure, stating that there is always sufficient number of channels to select with this computation:

$$TN_j = \frac{(M - t_j) \times S}{2 \times t_j} \tag{6}$$

where M is the total number of available channels, t_j is the number of channels required in a cell j and S is the size of the neighborhood. However, we have observed that the tabu tenures become greater than the maximum number of iterations when we use (6). This means that once an assignment is marked as tabu, it will never be considered again. As the iterations are carried further, more and more assignments are dropped, constraining the problem more than necessary. Therefore, we adopted the formulation of [3] with a slight modification:

$$TN_j = \frac{(M - t_j) \times S}{200 \times t_j} \tag{7}$$

For the LTM implementation, as in [3], we compare the residence frequency of a particular assignment with its frequency threshold and mark the assignment as tabu if its threshold value has been exceeded. The residence measure for an assignment determines the number of times that assignment belongs to the solution set over the total number of iterations so far. In other words, it determines the number of times that the value of a particular assignment (cell number and channel number pair) is 1 in the solution. The residence measure is denoted as follows [3]:

$$FM_{jk} = \frac{B_{jk}}{T} \tag{8}$$

where B_{jk} is the number of times the assignment belongs to solutions and T is the total number of iterations so far. Furthermore, the frequency threshold is selected as follows [3]:

$$TR_j = \frac{2 \times t_j}{(M - t_j) \times S} \tag{9}$$

where M is the total number of available channels, t_j is the number of channels required in cell j and S is the size of the neighborhood.

In our implementation, we considered three types of termination criteria. The first and most straightforward criterion is reaching a predetermined maximum number of iterations of the algorithm. However, the search may result in an optimum solution before the maximum number of iterations is reached. If an optimum solution is found, the search is terminated. There are many cases where the optimum solution cannot be found and the algorithm cannot make a significant improvement. If the number of iterations without a significant improvement reaches a certain value, then the search is terminated. This value is a parameter that affects the average number of iterations, which is also a performance criterion in addition to the objective minimization.

B. Reactive Tabu Search

The reactive tabu search is an enhancement to the classical tabu search, initially proposed by Battiti *et al.* in [2]. The reactive tabu search enhances the robustness of the classical scheme by adapting the tabu list size to the properties of the optimization problem.

Adapting the tabu tenure: In the classical approach, marking a move as a tabu is done with a statically determined value. This approach lacks the detection of cycling solutions. In the reactive tabu approach, all solutions that have been visited so far are stored in the memory, along with their corresponding number of iterations. When a new solution is found, it is compared with all past solutions. If a cycle is detected, then the tabu tenure of the cell in question is increased. This increase is done by multiplying the tabu tenure of the cell by a value greater than 1. This value is represented by the INC variable, where $INC > 1$. We modify the original tabu tenure update mechanism proposed by the authors in [2] as follows: If the resulting value is greater than the maximum iteration count of the tabu search algorithm (*maxIter*), we set the tabu tenure to the maximum iteration count. The motivation for this slight modification is because we observed that the tabu tenure values may get excessively large if this check is not included. Furthermore, making the tabu tenure more than the maximum iteration count is insensible. Hence, the tabu tenure update process is as follows:

$$TN_j = \max(TN_j \times INC, maxIter) \tag{10}$$

This way, certain moves are penalized with a greater tabu value, thus suppressing the occurrence of cycles. Storing all solutions in the memory is a costly operation, and hashing techniques are used for storing this information. However, in our case, we do not deal with very large problems and considering today's computer technology, the memory requirement is not overwhelming.

Slow Reduction (SR) Mechanism: According to the authors of [2], when exploring parts of the search space where the list size need not be large, the list size is decreased by a slower reduction mechanism. This is necessary because if the tabu tenure grows unnecessarily large so that it constrains the search more than necessary, a slow process needs to reduce the tabu

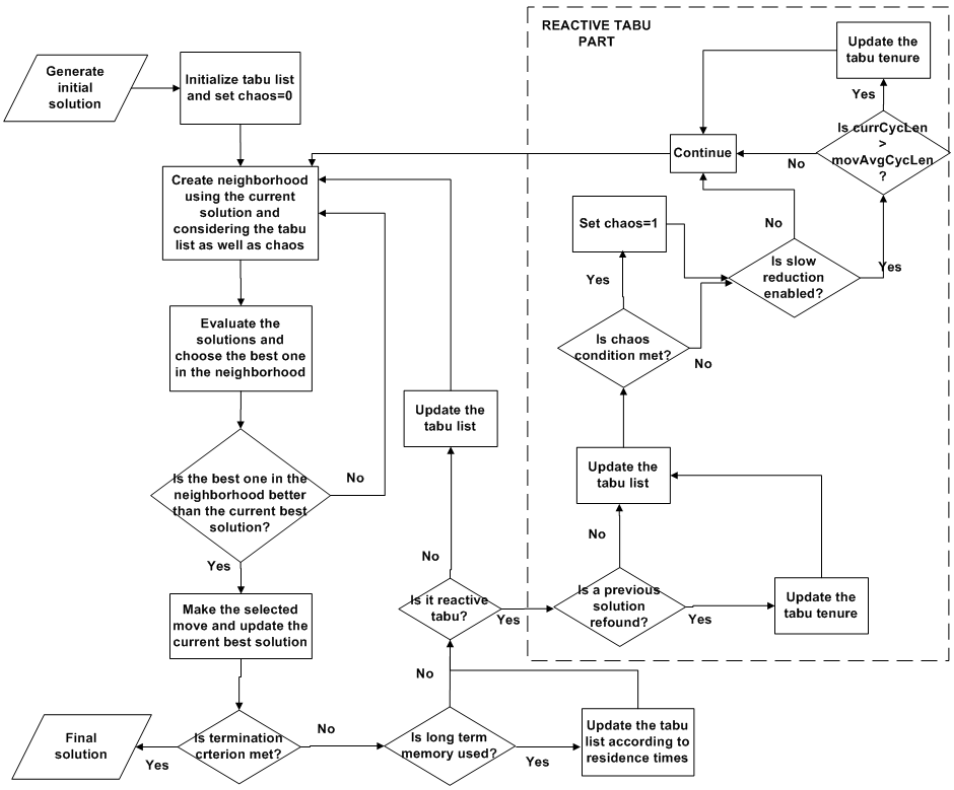

Fig. 1: The flowchart of our classical and reactive tabu search implementation.

tenure. When a cycle is detected, the moving average of cycle length is updated by:

$$\begin{aligned} movAvgCycLen &= (0.1 \times currCycLen) \\ &\quad +(0.9 \times movAvgCycLen) \end{aligned} \tag{11}$$

where *movAvgCycLen* denotes the current value of the moving averaged cycle length and *currCycLen* symbolizes the current cycle length. If a number of iterations greater than *movAvgCycLen* passed since the last tabu tenure size change, then the tabu tenure is decreased by multiplying the tabu tenure of the cell by a value less than 1. This value is represented by the DEC variable, where $DEC < 1$. Thus, the tabu tenure SR process is as follows:

$$\begin{aligned} &if(stepsSinceLastSizeChange < movAvgCycLen) \\ &\quad TN_j = TN_j \times DEC \qquad (12) \\ &end \\ &\quad stepsSinceLastSizeChange = 0 \qquad (13) \end{aligned}$$

Escape from Chaotic Traps: When exploring the search space, the search may get trapped in certain parts of the trajectory resulting in a frequent revisiting of certain solutions. This situation is referred to as a *chaotic attractor*. The authors in [2] suggest to alleviate this problem in two steps. The first step is randomly exchanging two units in the current solution and then executing the tabu search algorithm with this exchanged solution as the starting point. The number of times that this operation is implemented is determined by the moving average of the cycle length. For the second step, the authors choose the best of these solutions and continue the operations of the initial reactive tabu search.

We observe that iteratively running the tabu search is computationally expensive. Therefore, we propose a slight change to the move mechanism in our implementation. We conclude that a chaos situation has occurred when the detected cycle length is less than the chaos count. In our implementation, we observed the behavior of the solutions in chaotic traps, and selected the chaos count as 3. In other words, if a solution is repeated in less than 3 iterations, we invoke the escape mechanism. When a chaos situation occurs, the neighborhood search mechanism considers the exchange of channels for two cells, while ensuring that the channel demand of each cell is still satisfied. At the same time, the neighborhood search mechanism also considers the original move and takes the best solution among these two mechanisms. If none of these two mechanisms leads to a better solution than the current one, then one of the solutions that leads to the same objective function value as the current solution is selected as the next solution. We make this last modification in order to diversify the search mechanism. Moreover, when a chaos situation occurs, we do not take the tabu conditions of the solution assignments into account. Although not as computationally expensive as the chaos escape mechanism in [2], our solution still introduces some additional complexity. However, it is noteworthy to mention that we invoke this mechanism only when a chaos situation occurs. Figure 1 illustrates the classical and reactive tabu search mechanisms we implemented.

TABLE I: Methods Used

Notation	Method Name	Reactive	LTM	SR
M1	Classical	No	No	No
M2	Classical with LTM	No	Yes	No
M3	Reactive	Yes	No	No
M4	Reactive with SR	Yes	No	Yes
M5	Reactive with LTM	Yes	Yes	No
M6	Reactive with LTM and SR	Yes	Yes	Yes

V. EXPERIMENTAL RESULTS

In our work, we consider the co-channel interferences, which are also known as hard constraints. When trying to determine and improve the quality of our solutions, the first measure we take into account is the minimization of the objective function. However, besides the objective function, the number of iterations of the algorithm that leads to a final solution is also a measure of quality. The number of iterations at the end of the heuristic is affected by the maximum number of iterations allowed without a significant improvement in the objective function. Table I shows the notations assigned to each method used in the experiments, along with the presence of the reaction mechanism, LTM and the SR mechanism.

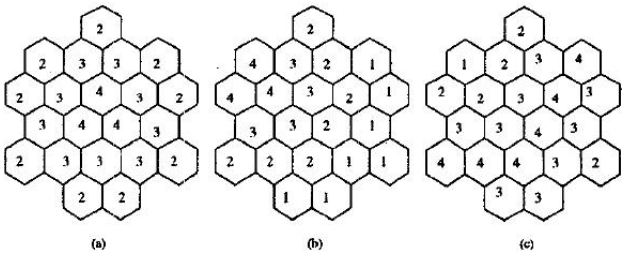

Fig. 2: Unbalanced traffic demands for the planar scenario [5].

A. Tabu Search vs. Genetic Algorithm

One of the works considering the hard constraints is by Jaimes-Romero *et al.* [5], which has planar scenarios with balanced and unbalanced traffic demands. In their work, the authors use genetic algorithms for the CAP. The scenario has 21 cells arranged planarly in the traditional hexagon fashion. The number of available channels is 12. The balanced traffic demands consist of 4 patterns, with each cell requiring c channels, where $c \in \{1, 2, 3, 4\}$. Figure 2 is from [5] and shows the three unbalanced traffic demand patterns.

For the implementation of both classical tabu search and the reactive tabu search, we used MATLAB. In the results shown in Table II and III, the maximum number of iterations is $N_{max} = 50$. The tabu search process terminates if the solution has not improved for $\lceil (N_{max}/4) \rceil = 13$ number of iterations or the optimum solution has been found; i.e., the objective value found equals zero. In reactive tabu, $INC = 1.5$ and $DEC = 0.9$. Tables II and III show the objective value; i.e., the number of constraint violations, as well as the average number of iterations among a total of 10 runs for each case. We also present the objective values obtained in [5]. The average number of iterations for the genetic algorithm denotes the generation number. In addition, we evaluate the performance of classical tabu without LTM , classical tabu with LTM, and reactive tabu with LTM, SR and chaotic escape mechanism.

From the results of the planar scenarios, we can deduce that in general, the family of reactive tabu search performs better on larger datasets. In the balanced channel demand scenarios, the objective and the number of iterations increase with the increasing demand. The best performance is obtained by the pure reactive tabu search, which is the third method. In the unbalanced channel demand scenarios, it is hard to report a single method that works for all types of demand scenarios; however, the reactive tabu search with LTM and SR (sixth method) seems to be performing better. Moreover, the results suggest that both tabu search families are more suitable for the second type of channel demand.

Note that the average number of iterations are close to the maximum number of iterations when the total channel demand is higher. This situation implies that the optimization process is terminated too early. Therefore, we present here another set of results for the planar scenario. In the results shown in Table IV and V, the maximum number of iterations is $N_{max} = 100$. The tabu search process terminates if the solution has not improved for $\lceil (N_{max}/4) \rceil = 25$ number of iterations or the optimum solution has been found; i.e., the objective value found equals zero. The remaining parameters of the tabu search are the same as in Tables II and III.

The results of a longer iteration show that the scenario with the high balanced demand and the second scenario with the unbalanced demand needed more iterations to converge. However, increasing the number of iterations also affects the second stopping criterion ('the solution is not improving.') This in turn worsens the results of the classical tabu search applied to low-demand scenarios.

B. Performance of the Tabu Search in reaching optimality

Another set of data is presented in [8]. This is a real life data obtained from a 24×21 km area in Helsinki, Finland. There are 25 cells, and the channel demands of the cells are unbalanced. This dataset is large enough for testing the performance of our approach in reaching optimality. The authors of [8] report that the optimal solution can be reached if there are 73 channels in total.

In our experiments, we set the total number of channels available to 73 and try to minimize the objective function, where the optimum is 0. The results in Table VI are obtained where the maximum number of iterations is $N_{max} = 200$. The tabu search process terminates if the solution has not improved for $\lceil (N_{max}/4) \rceil = 50$ number of iterations or the optimum solution has been found; i.e., the objective value found equals zero. The remaining parameters of the tabu search are the same as in Section V-A.

The scenario obtained from Finland resembles the planar scenario in Section V-A; i.e., the number of cells are close to each other. Nevertheless, the interference and channel demand constraints are much more challenging. Consequently, converging to the optimal solution requires more iterations. This time, the optimal solution is known to be 0, so that the performance (in terms of reaching optimality) of the tabu search mechanisms

TABLE II: Planar Scenario with Balanced Traffic Demands, $N_{max} = 50$

	Average Objective							Average number of iterations						
Demand Type	M1	M2	M3	M4	M5	M6	GA [5]	M1	M2	M3	M4	M5	M6	GA [5]
1	0	0	0	0	0	0	27	4.5	3	2.9	3.1	3.4	2.7	170
2	0	0	0	0	0	0	21	12.3	11.5	10.7	10.4	11.5	12.7	195
3	2.6	2.3	0.1	0.5	0.6	0.5	27.2	42.4	42.7	35.5	38.4	36.5	34.5	198
4	17.3	17.4	11.5	12.6	12.8	13.2	38.6	46.9	48.4	49.7	49.5	49.9	48.3	196

TABLE III: Planar Scenario with Unbalanced Traffic Demands, $N_{max} = 50$

	Average Objective							Average number of iterations						
Demand Type	M1	M2	M3	M4	M5	M6	GA [5]	M1	M2	M3	M4	M5	M6	GA [5]
1	5.3	5	2.1	2.8	2.3	1.7	29	40	40	43.6	41.7	44.8	43.2	175
2	2.2	2.9	1.4	0.6	1.1	0.6	25.4	31.6	31	27.7	25.6	27.4	27.6	186
3	4.8	5.3	1.8	2.1	1.9	1.9	28.8	41.1	42.5	43.5	45.1	48.5	42.9	192

TABLE IV: Planar Scenario with Balanced Traffic Demands, $N_{max} = 100$

	Average Objective							Average number of iterations						
Demand Type	M1	M2	M3	M4	M5	M6	GA [5]	M1	M2	M3	M4	M5	M6	GA [5]
1	0	0	0	0	0	0	27	3.1	3.4	2.9	3.7	3.3	3.8	170
2	0	0	0	0.1	0.1	0	21	12.2	11.9	12	13.2	14.9	11.2	195
3	3.3	3.5	0.2	0.3	0.2	0.5	27.2	51.8	51.5	32.8	40.6	39	39.9	198
4	15.6	17.2	8.9	10.7	8.9	8.1	38.6	68.4	65.2	87.6	87	97.8	91.7	196

TABLE V: Planar Scenario with Unbalanced Traffic Demands, $N_{max} = 100$

	Average Objective							Average number of iterations						
Demand Type	M1	M2	M3	M4	M5	M6	GA [5]	M1	M2	M3	M4	M5	M6	GA [5]
1	4.7	4.4	1.3	2.4	2.2	1.7	29	55.3	56.1	56.4	56.2	57.9	59.7	175
2	2	2.3	1	0.6	0.8	0.7	25.4	42.6	44.8	42.1	40.4	38.3	43.4	186
3	5.5	6.4	1.5	1.1	1.6	1.6	28.8	57.6	55.6	63.5	66.6	63.1	66.9	192

are observable. As expected, the family of reactive tabu search methods perform better than the classical tabu search family.

TABLE VI: Finland Scenario

Method	Average Objective	Average number of iterations
M1	4.6	159.7
M2	4.5	161.8
M3	2.8	139.4
M4	2.9	142.1
M5	3.1	135.0
M6	3.3	131.8

Note that in all test scenarios, the reactive tabu search with LTM (fifth and sixth methods) performs slightly worse than the reactive tabu search without LTM. The reason for this kind of behavior can be attributed to the overrestriction of the search space. In the reactive tabu search, using the INC variable increases the tabu tenures and therefore the penalties applied. Consequently, certain moves are excluded from the search space for a larger number of iterations. When the LTM is activated, more moves are left out and search space becomes too small to reach the optimal solution.

VI. CONCLUSION AND FUTURE DIRECTIONS

In this paper, we have applied the classical tabu search and the reactive tabu search to the channel assignment problem. Moreover, we explored the impacts of the LTM, reaction mechanism and SR mechanism. The results we obtained from two sets of scenarios show that the reactive tabu search performs better, especially in the cases where the channel demands are high. Furthermore, we have deduced that the use of an LTM together with the reaction mechanism overrestricts the search space and degrades the quality of the solutions.

For ease of implementation and simplicity, we have only considered the co-channel interference, which is a hard constraint. For a more realistic and broader approach, soft constraints such as the adjacent-channel and co-site interferences can also be considered in the problem formulation.

REFERENCES

[1] K. Aardal, S. van Hoesel, A. Koster, C. Mannino, and A. Sassano, "Models and solution techniques for frequency assignment problems," *Annals of Operations Research*, vol. 153, no. 1, pp. 79–129, 2007.

[2] R. Battiti and G. Tecchiolli, "The Reactive Tabu Search," *ORSA Journal on Computing*, vol. 6, pp. 126–140, 1994.

[3] Y. Peng, L. Wang, and B. Soong, "Optimal channel assignment in cellular systems using tabu search," in *Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003. 14th IEEE Proceedings on*, vol. 1, 2003.

[4] I. Blöchliger and N. Zufferey, "A graph coloring heuristic using partial solutions and a reactive tabu scheme," *Computers and Operations Research*, vol. 35, no. 3, pp. 960–975, 2008.

[5] F. Jaimes-Romero, D. Munoz-Rodriguez, and S. Tekinay, "Channel assignment in cellular systems using genetic algorithms," in *Vehicular Technology Conference, 1996.'Mobile Technology for the Human Race'., IEEE 46th*, vol. 2.

[6] R. Montemanni, D. Smith, and S. Allen, "Lower Bounds for Fixed Spectrum Frequency Assignment," *Annals of Operations Research*, vol. 107, no. 1, pp. 237–250, 2001.

[7] N. Funabiki and Y. Takefuji, "A neural network parallel algorithm for channel assignment problems in cellular radio networks," *Vehicular Technology, IEEE Transactions on*, vol. 41, no. 4, pp. 430–437, 1992.

[8] D. Kunz, "Channel assignment for cellular radio using neural networks," *IEEE Transactions on Vehicular Technology*, vol. 40, no. 1 Part 2, pp. 188–193, 1991.