

Solving the Radio Link Frequency Assignment Problem using Guided Local Search

Christos Voudouris
Intelligent Systems Research Group
MLB 1/PP 12
BT Laboratories
Ipswich IP53RE
United Kingdom
e-mail: chrisv@info.bt.co.uk

Edward Tsang
Department of Computer Science
University of Essex
United Kingdom
e-mail: edward@essex.ac.uk

1. SUMMARY

In this paper, we examine the application of the combinatorial optimisation technique of Guided Local Search to the Radio Link Frequency Assignment Problem (RLFAP). RLFAP stems from real world situations in military telecommunications and it is known to be an NP-hard problem. Guided Local Search is a metaheuristic that sits on top of local search procedures allowing them to escape from local minima. GLS is shown to be superior to other methods proposed in the literature for the problem, making it the best choice for solving RLFAPs.

2. INTRODUCTION

Guided Local Search (GLS) is a general optimisation technique suitable for a wide range of combinatorial optimisation problems [21]. Successful applications of the technique so far include practical problems such as Workforce Scheduling [20], Frequency Assignment [22] and Vehicle Routing [12]. Classic problems such as the Travelling Salesman Problem (TSP) and the Quadratic Assignment Problem (QAP) have also been tackled with the method. GLS has been shown to be equally good if not better than the best TSP and QAP heuristic search algorithms [23, 21].

GLS was derived from the GENET neural network [25] for Constraint Satisfaction Problems [19] and extends the approach used in GENET to the bulk of combinatorial optimisation problems. GLS belongs to a class of techniques known as *Metaheuristics*. Prominent members of this class include *Tabu Search* [8], *Simulated Annealing* [11], *Genetic Algorithms* [15], and others. Metaheuristics aim at enhancing the performance of heuristic methods in solving large and difficult combinatorial optimisation problems. The method can be used alone or in conjunction with Fast Local Search (FLS). Fast Local Search is a neighbourhood search reduction scheme that particularly suits the operations of GLS [22, 20, 23].

The Radio Link Frequency Assignment Problem (RLFAP) considered in this article is a member of a class of problems known as Frequency Assignment Problems (FAPs). FAPs are NP-hard and they are closely related to the well-known Graph Colouring problem. For a very recent and comprehensive survey of the different FAP variants and problem solving strategies, the reader may refer to [14].

RLFAP stems from real world problems in military telecommunications. A number of RLFAP instances have been released in the framework of the CALMA (Combinatorial ALgorithms for Military Applications) project. The project involved six European research groups and one of its major objectives was to develop and test algorithms for frequency assignment problems. For an overview of the research contacted in the framework of the CALMA project the reader may refer to [18]. FAPs in general and RLFAP in particular are dealing with the allocation of a limited radio spectrum resource to a number of users in an optimal way.

Guided Local Search and its particular application to the RLFAP described in this paper can be easily extended to other variations of the FAP. In general, the technique is directly applicable to a wider class of problems, known as Partial Constraint Satisfaction Problems [19]. PCSPs extend the classic Constraint Satisfaction Problem (CSP) problem [19] to cover over-constrained instances where no solution exists which satisfies all the constraints. In such cases, we are often seeking solutions which violate the minimum number of constraints.

PCSPs can be used to model a variety of problems where the ever-increasing demand for a limited resource leads to situations where constraints on service quality can be softened. In the context of frequency assignment, constraints that transmitters should not interfere with each other need to be softened because of the limited radio spectrum available, which makes interference unavoidable.

For a detailed description of GLS for PCSPs, the reader may refer to [22, 21]. In this work, we focus on variants of the GLS algorithm for the RLFAP. The paper is structured as follows. We first describe the RLFAP. Following that, we present Local Search, Fast Local Search and Guided Local Search procedures for the problem. The different variants are evaluated on publicly available RLFAP benchmarks and results are compared with a variety of other methods that have been applied to the same instances. The comparisons made clearly indicate the superiority of GLS over alternative approaches proposed for the problem.

3. THE RADIO LINK FREQUENCY ASSIGNMENT PROBLEM

The *Radio Link Frequency Assignment Problem* (RLFAP) was abstracted from the real life application of assigning frequencies to radio links. The interference level between the frequencies assigned to the different links has to be acceptable; otherwise communication will be distorted. The frequency assignments have to comply with certain regulations and physical characteristics of the transmitters. Moreover, the number of frequencies is to be minimised, because each frequency used in the network has to be reserved at a certain cost. In certain cases, some of the links may have pre-assigned frequencies which may be respected or preferred by the frequency assignment algorithm. RLFAP is NP-Hard and it is a variant of the graph T-colouring problem as introduced by Hale [9]. IP formulations of the problem can be found in [14]. Following [18], the problem can be briefly described as follows.

We are given a set L of links. For each link i a frequency f_i has to be chosen from a given domain D_i . Some links may already have a pre-assigned frequency pf_i , which may or may not be changed. Constraints are defined on pairs of links that limit the choice of frequencies for these pairs. For a pair of links $\{i, j\}$ these constraints are either of type

$$|f_i - f_j| > d_{ij} \quad (1)$$

or of type

$$|f_i - f_j| = d_{ij} \quad (2)$$

for a given distance $d_{ij} \geq 0$. Two links i and j involved in a constraint of type (1) are called *interfering* links, and the corresponding d_{ij} is the interfering distance. Two links bound by a constraint of type (2) are referred to as a pair of *parallel* links; every link belongs to exactly one such pair.

Some of the constraints may be violated at a certain cost. Such restrictions are called *soft*, in contrast to the hard *constraints*, which may not be violated. The constraints of type (2) are always hard, and so are some of the pre-assignment constraints which dictate the preferred frequency for a link also called *mobility constraints*. Interference costs c_{ij} for violating soft constraints of type (1) and mobility costs m_i for changing soft pre-assigned frequencies are given. An assignment of frequencies is complete if every link in L has a frequency assigned to it. We denote by C and M the sets of all soft *interference* and *mobility* constraints, respectively.

The first priority is to find a complete assignment that satisfies all hard constraints and is of minimum cost:

$$\min_C \sum_{c_{ij}} \delta(|f_i - f_j| \leq d_{ij}) + \sum_M c_i \delta(|f_i - pf_i| > 0) \quad (3)$$

subject to hard constraints:

$|f_i - f_j| > d_{ij}$: for all pairs of links $\{i, j\}$ involved in the hard constraints,

$|f_i - f_j| = d_{ij}$: for all pairs of parallel links $\{i, j\}$,

$f_i = p_i$: for all links $i \in L$ with hard pre-assigned frequencies,

$f_i \in D_i$: for all links $i \in L$,

where $\delta(\cdot)$ is 1 if the condition within brackets is true and 0 otherwise.

If there exists a feasible assignment (i.e. a complete assignment of zero cost) then we would like to find a feasible

assignment that satisfies all constraints and minimises the number of distinct frequencies used or the largest frequency used:

$$\min |\cup_i \{f_i\}| \quad (4)$$

or

$$\min \max_i f_i \quad (5)$$

subject to the hard and soft constraints:

$|f_i - f_j| > d_{ij}$: for all pairs of links $\{i, j\}$ involved in the soft and hard constraints,

$|f_i - f_j| = d_{ij}$: for all pairs of parallel links $\{i, j\}$,

$f_i = p_i$: for all links $i \in L$ with hard pre-assigned frequencies,

$f_i \in D_i$: for all links $i \in L$.

Essentially, there are two distinct types of RLFAP instances. Instances where all solutions violate one or more constraints of the problem. For these *insoluble* instances, the cost function is given by (3). Instances for which exist at least one solution which satisfies all the constraints. For these *soluble* instances, we are seeking solutions which satisfy the constraints and either minimise (4) or (5). In the following, we describe the proposed GLS algorithm for the problem. We begin with local search which is the basis of the method.

4. LOCAL SEARCH FOR THE RLFAP

A local search procedure for PCSPs which is also applicable to the RLFAP has been described in previous work of the authors [22]. The scheme is based on the min-conflicts heuristic of Minton et al. [13] for Constraint Satisfaction Problems and also the computational model of the GENET neural network [25, 6]. An 1-optimal type move is used which changes the value of one variable at a time. Starting from a random and complete assignment of values to variables, variables are examined in an arbitrary static order. Each time a variable is examined, the current value of the variable changes to the value (in the variable's domain) which yields the minimum value for the objective function. Ties are randomly resolved allowing moves which transit to solutions with equal cost. These moves are called *sideways moves* [17] and enable local search to examine plateau of solutions occurring in the landscapes of many SAT and CSP problems.

The above local search procedure can be directly applied to RLFAP by considering each link as a variable with a domain given by all the possible frequencies that can be assigned to the link. Results on using this approach were reported in [22]. A more efficient approach, used in this work, takes into account the fact that each link in RLFAP is connected to exactly one other link via a hard constraint of type (2). In particular, we can define a local search variable for each pair of parallel links bound by an equality constraint [7]. The domain of this variable is defined as the set of all pairs of frequencies from the original domains of the parallel links which satisfy the hard equality constraint. Next, we examine how this basic local search procedure can be improved by taking advantage of the Fast Local Search approach.

5. FAST LOCAL SEARCH FOR THE RLFAP

Best improvement local search for the RLFAP as used in the context of Tabu Search [1, 7] evaluates all possible 1-optimal moves over all variables before selecting and performing the best move. Given the large number of links in real world instances, greedy local search is a computationally expensive option. The local search procedure described in previous section is already a faster alternative. An even faster alternative is fast local search initially used in the TSP under the name *don't look bits* technique for solving large instances of the problem [2]. Fast local search has been abstracted by Voudouris and Tsang and used in a number of applications as a generic neighbourhood reduction scheme [22, 20, 23].

In the case of the RLFAP, a bit is attached to each variable. If the bit of the variable is 1 then the variable is called *active* and it is examined for improving moves otherwise it is called *inactive* and it is discarded by local search. Whenever a variable is examined and its value is changed (i.e. the variable's parallel links are assigned to another pair of frequencies because of an improving or sideways move) the activation bit of the variable remains to 1 otherwise it turns to 0 and the variable is excluded in future iterations of the improvement loop. Additionally, if a move is performed activation spreads to other variables which have their bits set to 1. In particular, we set to 1 the bit of variables where improving moves may occur as a result of the move just performed. These are the variables for which either one of their links is connected via a constraint to one of the links of the variable that changed value. There are five potential schemes for propagating activation after changing the value of a variable. They are the following:

- S1. Activate all variables connected via a constraint to the variable which changed value.
- S2. Activate only variables that are connected via a constraint which is violated. This resembles CSP local search methods where only variables in conflict have their neighborhood searched.
- S3. Activate only variables that are connected via a constraint which has become violated as a result of the move (subset of S2 and also S4).
- S4. Activate only variables that are connected via a constraint that changed state (i.e. violated \rightarrow satisfied or satisfied \rightarrow violated) as a result of the move (superset of S3).
- S5. Activate variables that fall under either condition S2 or S4.

Fast local search stops when all the variables are inactive or when a local minimum is detected by other means. In the following, we describe the metaheuristic technique of GLS which significantly improves the performance of local searches.

6. GUIDED LOCAL SEARCH

Guided local search is a metaheuristic algorithm which its aim is primarily to help local search to escape local optima. In doing so, the algorithm is also trying to distribute the search efforts to the best effect by guiding local search towards solutions of high quality. The basic idea is to augment the objective function with penalties, which direct the search away from local optima and towards high quality solutions.

To apply GLS, one has to define *features* for the candidate solutions. For example, in the travelling salesman problem, a feature could be "whether the candidate tour travels immediately from city A to city B" [23]. GLS associates *costs* and *penalty* to each feature. The costs should normally take their values from the objective function. For example, in the travelling salesman problem, the cost of the above feature is the distance between cities A and B. The penalties are initialised to 0 and will only be increased when the local search reaches local optimum. This will be elaborated below.

Given an objective function g that maps every candidate solution s to a numerical value, we define a function h which will be used by local search (replacing g).

$$h(s) = g(s) + \lambda \times \sum(p_i \times I_i(s)) \quad (6)$$

where s is a candidate solution, λ is a parameter to the GLS algorithm, i ranges over the features, p_i is the penalty for feature i (all p_i 's are initialised to 0) and I_i is an indication of whether s exhibits feature i :

$$I_i(s) = 1 \text{ if } s \text{ exhibits feature } i; 0 \text{ otherwise} \quad (7)$$

When local search settles on a local optimum, the penalty of some of the features associated to this local optimum is increased (to be explained below). This has the effect of changing the objective function (which defines the "landscape" of the local search) and driving the search towards other candidate solutions. The key to the effectiveness of GLS is in the way that penalties are imposed. It is worth pointing out that a slight variation in the way that penalties are managed could make all the difference to the effectiveness of a local search.

Our intention is to penalise "bad features", or features which "matter most", when a local search settles in a local optima. The feature that has high cost affects the overall cost more. Another factor that should be considered is the current penalty value of that feature. We define the utility of penalising feature i , *util_i*, under a local optimum s_* , as follows:

$$Util_i(s_*) = I_i(s_*) \times c_i / (1 + p_i) \quad (8)$$

where c_i is the cost and p_i is the current penalty value of feature i . In other words, if a feature is not exhibited in the local optimum, then the utility of penalizing it is 0. The higher the cost of this feature (c_i), the greater the utility of penalising it. Besides, the more times that it has been penalised, the lower the utility of penalising it again.

In a local optimum, the feature(s) with the greatest *util* value will be *penalised*. This is done by incrementing its penalty value by 1:

$$p_i = p_i + 1 \quad (9)$$

By taking cost and the current penalty into consideration in selecting the feature to penalise, we are distributing the search effort in the search space. Candidate solutions which exhibit "good features", i.e. features involving lower cost, will be given more effort in the search, but penalties help to prevent all effort be directed to the best features. Following we shall describe the general GLS procedure:

Procedure GLS (input: an objective function g ; a local search strategy L ; features and their costs; parameter λ)

0. Generate a starting candidate solution randomly or heuristically;

1. Initialize all the penalty values (p_i) to 0;

2. Repeat the following until a termination condition (e.g. a

maximum number of iterations or time limit) has been reached:

3.1. Perform local search (using L) according to the function h (which is g plus the penalty values, as defined in (6) above) until a local optimum M has been reached;

3.2. For each feature i which is exhibited in M compute $util_i = c_i / (1 + p_i)$

3.3. Penalize every feature i such that $util_i$ is maximum: $p_i = p_i + 1$;

Return the best candidate solution found so far according to the objective function g .

Applying guided local search to a problem requires the existence of a local search procedure preferably a version of fast local search and also a set of features which will be used to guide local search in the problem's search space. If a fast local search variant is available, we can combine it with GLS in a straightforward way. The key idea is to associate features to activation bits. The associations to be made are such that for each feature we know which variables contain moves that have an immediate effect upon the state of the feature (i.e. moves that remove the feature from the solution).

At the beginning of GLS, all the activation bits of fast local search are set to 1 and fast local search is left to reach the first local minimum. Whenever a feature is penalised, the bits of the associated variables and only these are set to 1. In this way, after the first local minimum, fast local search calls start by examining a subset of the variables and in particular those which associate to the features just penalised.

Another variant to the GLS method which seems to significantly improve performance in certain RLFAP instances is to decrease penalties and not only increase them. More specifically, the variant uses a circular list to retract the effects of penalty increases made earlier in the search process, in a way that very much resembles a tabu list. In particular, penalties increased are decreased after a certain number of penalty increases is performed. The scheme uses an array of size t where the t most recent features penalised are recorded. The array is treated as a circular list, adding elements in sequence in positions 1 through t and then starting over at position 1. Each time the penalty of a feature is increased (by one unit), the feature is inserted in the array and the penalty of the feature previously stored in the same position is decreased (by one unit). The rationale behind the strategy is to allow GLS to return to regions of the search visited earlier in the search process, so introducing a search intensification mechanism.

Next, we examine the objective function g used by local search in the RLFAP, the GLS features defined, their costs and also how the problem variables are activated in the case of fast local search when these features are penalised.

7. LOCAL SEARCH AND GLS APPLIED TO THE RLFAP

7.1 Objective Function

In the RLFAP, we defined and used a simple objective function for both insoluble and soluble instances. The objective function g was given by the sum of all constraint violation costs in the solution with all the constraints contributing equally to the sum. This objective function is as follows:

$$g(s) = \sum_{C \cup C^{\text{Hard}}} \delta(|f_i(s) - f_j(s)| \leq dij) + \sum_M \delta(|f_i(s) - pf_i| > 0) \quad (10)$$

subject to hard constraints:

$$f_i(s) \in D'_i : \text{for all links } i \in L,$$

where $\delta(\cdot)$ is 1 if the condition within brackets is true and 0 otherwise, $f_i(s)$ is the frequency assigned to link i in solution s , C^{Hard} is the set of hard inequality constraints, C is the set of soft inequality constraints, M is the set of soft mobility constraints and D'_i is the reduced domain for link i containing only frequencies which satisfy the hard equality and mobility constraints. A solution s with cost 0 with respect to g is satisfying all hard and soft constraints of the problem.

GLS allows us to use such a simple objective function for local search and still obtain high quality solutions. Most of the costs contained in the original cost function of the problem can be minimised by defining features for solutions and setting the feature costs to appropriate values. The application of penalties can force local search toward solutions of high quality, to some degree independently from the objective function used by local search.

GLS will normally perform better with an objective function which is closely based on the cost function of the problem. The motivation to use a simple function such as (10) is closely related to the rugged landscapes formed in RLFAP, if the original cost function is used. In particular, high and very low violation costs are defined for some of the soft constraints in insoluble instances. This leads to even higher violation costs to have to be defined for hard constraints. The landscape is not smooth but full of deep local minima mainly due to the hard and soft constraints of high cost. Soft constraints of low cost are buried under these high costs. CALMA researchers, implementing Simulated Annealing for the insoluble instances, also found it difficult to devise a cooling regime which is effective on these types of landscapes [7].

In the following, we continue by presenting the different features defined in the RLFAP to help local search escape local minima and also be guided towards solutions of high quality.

7.2 Constraint Features for Minimising Interference Costs

One very important cost factor in the RLFAP is the constraint violation costs defined for soft inequality constraints. Inequality constraints can be used to define a basic feature set for the RLFAP. Each inequality constraint is interpreted as a feature with the feature cost given by the violation cost of the constraint c_{ij} as defined in the problem's original cost function (3).

For the RLFAP, the objective function of local search is augmented with a set of modifiable penalty parameters one for each inequality constraint. Initially, the penalty parameters are set to 0 and each constraint accounts only for its violation cost set to 1 for all constraints. Each time local search settles in a local minimum, the penalties for some of the constraints violated (the corresponding features are exhibited) are increased according to the general scheme described in section 6. Constraints with high cost are penalised more frequently than those with low cost. In the short term, local search escapes from the local minimum while in the long term, it is biased to spend more time on solutions that satisfy high cost constraints than low cost ones.

Hard inequality constraints are also modelled as features though the cost assigned to them is infinity. This results in their utility to be penalised to also tend to infinity. To implement that in the program, hard constraints are given priority over all other features (i.e. features defined for soft inequality constraints and other features to be introduced below). This basically forces local search to return back to a feasible region where penalising other features can resume.

7.3 Value Features for Minimising Mobility Costs

The mobility constraints considered during the search process are all of soft nature since the hard mobility constraints are pre-processed. To minimise this cost factor, a feature is defined for each value in the domain of a variable and that for all variables which have mobility constraints associated with their links. The cost of each *value feature* is defined by the sum of mobility costs m_i that would be incurred according to (3) if the pair of frequencies represented by the value were assigned to the links modelled by the variable.

Each time local search settles in a local minimum, values assigned to variables with mobility constraints or constraints violated (i.e. the corresponding features are exhibited) will have their penalties increased according to the general scheme described in section 6. If GLS is combined with the fast local search of section 5 then variables associated with the penalised constraints or values will also be activated to focus fast local search to remove the penalised features.

7.4 Frequency Features for Minimising the Number of the Frequencies Used

Features for minimising this factor are only deployed in soluble instances. In particular, a *frequency feature* is defined for each frequency appearing in the union of domains of the links. In contrast to other features described above, the penalty for a frequency feature is incorporated in the objective function not once but multiple times. More specifically, for each variable that can use a specific frequency, the penalty for this frequency is incorporated in the objective function multiplied by an indication function. This indication function takes the value 1, if the frequency is utilised by the variable in at least one of its parallel links or 0 otherwise. This enable us to force all variables that are using a specific frequency to avoid it by simply increasing the penalty for that frequency. Given the capability to force local search to avoid specific frequencies, the question is how to choose these frequencies so that the overall number of frequencies used is minimised during the search process.

A simple heuristic is to increase the penalties for frequencies used by only few of the variables. Eliminating these frequencies will eventually lead to a decrease in the total number of frequencies used since variables will be in their majority assigned to frequencies used by many variables. More formally, the feature cost for a feature representing frequency $f \in \cup_i \{f_i\}$ is defined as follows:

$$c_f(s) = 1/|L_f(s)| \quad (11)$$

where L_f is the set of links assigned to the frequency f in solution s . Obviously, frequencies not assigned to any of the links in the local minimum may be discarded since penalising them will have no effect. The indication function for a frequency feature to be incorporated in its utility function is as follows:

$$I_f(s_*) = 1 \text{ if } s_* \text{ uses frequency } f; 0 \text{ otherwise.} \quad (12)$$

Frequency features maximising the utility function (8) (as it is instantiated by using (11) and (12)) have their penalties incremented.

If GLS is combined with fast local search, we activate all variables which are using the penalised frequency in either one of their links. Finally, frequency features are only considered for penalisation if no soft or hard constraints are violated since we want to minimise these cost factors first.

7.5 Value Features for Minimising the Maximum Frequency Used

For minimising this optimisation criterion, we introduced a feature set similar to that used for mobility costs. As with mobility costs, one feature was defined for each value in the domain of a variable though this time the feature costs were all equal and set to the value 1. For the penalties on values to have an effect in minimising the maximum frequency used, we only considered for penalising values which were responsible for assigning the maximum frequency used in the local minimum. Since all features are having an equal cost, the utility function of GLS simply translates in selecting the feature(s) which has been penalised the minimum number of times. As before, if fast local search is utilised, the variable of a value penalised is activated.

8. EXPERIMENTAL EVALUATION OF GLS

8.1 Experimental Setting

We conducted a large number of experiments on the publicly available RLFAP instances. For all GLS variants tested, the λ parameter was set to 1 and ten runs were performed from different random initial solutions for each benchmark instance. Since GLS is not a complete method, a termination criterion has to be specified. For that purpose, we set a limit on the number of 1-optimal moves that are evaluated in a single run. The value chosen for that limit was 4×10^7 moves for all variants and instances considered.

In order to give an indication of the times spent on problem solving, we also measured the *User Time* (excluding any *System Time* such as i/o operations, memory allocation etc.) required by the variants to reach the best solution in each of their runs. All times reported refer to a SPARCUltra workstation running at 168Mhz which represents a typical modern UNIX workstation.

To evaluate the performance of variants with regard to the quality of solutions they can produce, we used the measure of percentage **excess** above the best known solution. This measure is calculated in the following way:

$$\text{excess} = 100 * (\text{solution_cost} - \text{best_known_solution_cost}) / \text{best_known_solution_cost}$$

In total, we used 25 instances to test GLS and its variants. Eleven of these instances were made publicly available by the French Centre d' Electronique l' Armement. These instances are based on real world problems and are considered to be representative of the different variations of the problem. We will refer to these instances as the *Celar* set.

In addition to these instances, a set fourteen were made available during the CALMA project. These instances were generated by the GRAPH problem generator described in [3].

Although random in nature, they thought to resemble the realistic Celar instances. We will refer to this second set of instances as the *Graph* set. In the following, we highlight some of the results from the experiments conducted. A detailed report on the results and the also the GLS method on the RLFAP will be provided in [24].

8.2 Guided Local Search Variants for the RLFAP.

In total, there are six variations of GLS for the RLFAP problem. The variants are derived by combining GLS with the local procedures presented in sections 4 and 5. These variants are the following:

- GLS-FI: This is the result of the combination of GLS with first improvement local search of section 4.
- GLS-FLS-S1, GLS-FLS-S2, GLS-FLS-S3, GLS-FLS-S4, GLS-FLS-S5: This is the set of variants resulting from the combination of GLS with fast local search using one of the available activation schemes (see section 5).

We tested all the above techniques on the RLFAP instances and also experimented with the circular list strategy when combined with GLS-FLS-S3, GLS-FLS-S4, and GLS-FLS-S5.

8.3 Best Known Solutions and GLS

GLS variants repeatedly found or improved the best known solutions as reported in [18] and also in an Internet Web page by Thomas Schiex devoted to the RLFAP instances (<http://www-bia.inra.fr/T/schiex/Doc/CELARE.html>). Table 1 summarises this information. In this table, solution costs in italic characters indicate that GLS was able to find the best known solution, while in bold indicate that GLS improved the best known solution. An asterisk next to a solution cost indicates that the cost is known to be optimal.

For the 25 publicly available RLFAP instances, GLS managed to find the best known solution in 18 of them, improved the best known solution in 5 and found a marginally inferior solution in only 2 of the instances. Note here, that the best known solutions have been discovered by different techniques. GLS achieved a better performance than the collection of these techniques by improving the cost of the previously best known solutions by an average 1,75%.

8.4 Comparing the GLS Variants.

All the GLS variants achieved very good results. It is worth pointing that despite the thousands of runs performed for testing the 6 variants, GLS was always able to find a feasible solution for the soluble instances and that despite the fact that hard inequality constraints were included in the cost function.

As expected, GLS-FLS variants were better on average than the GLS-FI variant. GLS-FLS-S5 was found to be the best variant with regard to the measures of average excess and worst excess. We also found that all GLS-FLS variants with the exception of GLS-FLS-S1 are well suited for finding high quality solutions (i.e. they have an average *best excess* between 1% and 2% above the best known solutions). GLS-FLS-S1 is activating more variables than the other FLS variants, this seems to be having an adverse effect in computation times which in each turn affects the quality of solutions found by GLS-FLS-S1. The variants considered in this section are not using the circular list strategy for penalties.

As we are going to see in the next section, this strategy can significantly improve the GLS performance.

RLFAP Instance	Best Known Solution Cost	Best Solution Cost found by GLS
celar1	16*	<i>16</i>
celar2	14*	<i>14</i>
celar3	14*	<i>14</i>
celar4	46*	<i>46</i>
celar5	792*	<i>792</i>
celar6	3389*	<i>3389</i>
celar7	343592	343598
celar8	262	262
celar9	15571	<i>15571</i>
celar10	31516	<i>31516</i>
celar11	22*	<i>24</i>
graph1	18*	<i>18</i>
graph2	14*	<i>14</i>
graph3	380*	<i>380</i>
graph4	394*	<i>394</i>
graph5	221	221
graph6	4189	4123
graph7	4324	<i>4324</i>
graph8	20	18
graph9	18*	<i>18</i>
graph10	394*	<i>394</i>
graph11	3513	3081
graph12	11827	<i>11827</i>
graph13	11130	10119
graph14	10	8*

Table 1 Comparison between the best known solutions and the best solutions found by GLS.

8.5 Results Using the Circular List Strategy

GLS variants can be further improved by utilising the circular list strategy described in section 6. We selected the variants GLS-FLS-S5, GLS-FLS-S4 and GLS-FLS-S3 to experiment with this strategy. GLS-FLS-S4 produced the best results when combined with the strategy while GLS-FLS-S5 and GLS-FLS-S3 also improved their performance. Results for GLS-FLS-S4 using using the circular list strategy are reported in Table 2. As we can see in this table, GLS-FLS-S4 achieves a stunning *average excess* of only 1.62% above the best known solutions. This is a massive improvement over the best *average excess* of 8,83% for GLS variants not using the circular list, achieved by GLS-FLS-S5 in the experiments of section 8.4.

Another result worth noting in Table 2 is that the best known solutions were found in each and every run of GLS-FLS-S4 for eleven of the instances (see rows with mean excess equal to 0). The average user time for the variant was a reasonable 4.73 minutes. Of course when used in practice, the algorithm can be stopped at any time and report the best solution found up to that point.

RLFAP Instance	Best Solution Cost	Worst Solution Cost	Mean Excess (%)	Mean User Time (min)
celar1	16	18	3.75	5.63
celar2	14	14	0	0.02
celar3	14	14	0	0.19
celar4	46	46	0	0.01
celar5	792	792	0	0.09
celar6	3389	3415	0.312777	15.16
celar7	343702	354205	0.442181	17.55
celar8	262	285	3.396947	15.68
celar9	15571	15612	0.04046	6.62
celar10	31516	31517	0.000635	5.46
celar11	24	30	22.72727	11.55
graph1	18	18	0	1.70
graph2	14	14	0	0.04
graph3	380	380	0	0.08
graph4	394	394	0	0.37
graph5	221	221	0	0.48
graph6	4133	4167	0.603929	11.84
graph7	4329	4338	0.157262	6.77
graph8	18	18	0	4.33
graph9	18	20	1.111111	1.67
graph10	394	764	9.390863	0.59
graph11	3087	3137	1.194417	4.67
graph12	11832	11840	0.049886	3.84
graph13	10194	11334	5.691274	3.92
graph14	8	8	0	0.08
Mean			1.627129	4.73

Table 2 Results for the variant GLS-FLS-S4 using the circular list strategy.

8.6 Comparison with the CALMA Project Algorithms.

In the framework of the CALMA project, a number of optimisation techniques were applied to the RFLAP, amongst them versions of Simulated Annealing, Tabu Search and Genetic Algorithms. A summary of the results for the techniques tested on the RFLAP by the CALMA project members is given in [18]. To give an indication of the relative performance of GLS with regard to these techniques, we compare the results reported in [18] with the results obtained by GLS-FLS-S4 using the circular list strategy. Tables 3 & 4 compare GLS results with the results from the CALMA techniques on the Celar and Graph sets respectively.

A number of conclusions can be drawn based on the information presented in these tables. Firstly, GLS applies to all instances without exception something which is not the case for any of the other algorithms. More importantly the performance of GLS is consistent across all instances. The only techniques which offer a wide coverage of instances and produce results consistent and competitive with those of GLS are the Simulated Annealing variant by EUT [7], the Variable Depth Search variant by EUT [7], and the Extended GENET by KCL which is a neural network [16, 1].

The results for Extended GENET are exceptionally good for a neural network architecture applied to a CO problem.

Nonetheless, they are not surprising since the technique is also based on the GENET neural network architecture like GLS. The interested reader may refer to [22] for a direct comparison between Extended GENET and the less efficient version of GLS presented in there. In brief, we found in [22] that the average solution quality produced by GENET is much inferior than that of GLS as implemented in that paper. The main reason being that Extended GENET does not use some of the principles of GLS such as selective penalisation and problem modelling using features. Furthermore, Extended GENET does not apply to problems with mobility costs and it is cumbersome to extend since a problem definition has to be cast to a neural network architecture.

The Simulated Annealing variant by EUT is performing well on both the soluble and insoluble instances of the CELAR set though overall the results are inferior in terms of solution quality than those of GLS. Furthermore the algorithm is very time intensive on the insoluble instances. This last behaviour may be related to the fact that it is difficult to find a good cooling regime for the rugged landscapes of RLFAP's insoluble instances as this is pointed out in [7].

In our opinion, the Variable Depth Search by EUT is the most competitive algorithm to the GLS technique though on average has an inferior performance. In particular, GLS has a significantly better performance on the insoluble instances of the Graph set and also the soluble instances of the CELAR set. A better solution by VDS is only reported for instances Celar7 and Graph7 and that by a very small margin. Furthermore the technique is not applied to instances where the maximum frequency is minimised.

Generally speaking, if the average behaviour of the CALMA methods was well documented then further conclusions could be drawn. As mentioned above, GLS-FLS-S4 has a very small variance with regard to solution quality (see Table 2) and that could be a significant advantage over other heuristic methods which frequently, they are very sensitive to the random starting point chosen in each run.

Finally, Tabu Searches (TS) developed by the CALMA project members are not very competitive, the main reason being the use of best improvement local search which given the large number of variables is a very expensive option for these problems. Further work on Tabu Search variants using candidate list strategies and/or move update schemes may result in better performance for Tabu Searches in the RLFAP. Successful TS variants have already been developed though in the context of the relatively simpler minimum interference problem [4, 5, 10]. RLFAP introduces a number of additional requirements such as mobility constraints, hard inequality and equality constraints, soft constraints with different violation costs, minimisation of the number of frequencies used, minimisation of the maximum frequency used, and also different domains for the problem's variables. At the moment, efficient TS approaches such as those described in [4, 5, 10] are only addressing the minimisation of constraints with equal violation costs and to some extent the problem of minimising the number of frequencies used though all variables are considered to have the same domain and the minimisation of the criterion is conducted in a rather ad-hoc way (i.e. multiple runs with fixed frequency sets [10]).

9. CONCLUSIONS

In this paper, the application of Guided Local Search to the Radio Link Frequency Assignment Problem was examined. A

number of different variants were described and evaluated on publicly available RLFAP benchmark instances. We found that the GLS method is an ideal technique for the problem finding solutions of high quality in reasonable time while being applicable to all different variations of the problem. In addition to that, comparisons of GLS with other methods demonstrated the superiority of the technique.

Given the very good performance of Guided Local Search on the problem (i.e. the best GLS variant achieves an average excess of 1,62% over the best known solutions for the benchmark instances), we believe there is little room for improvement with regard to solution quality by using more sophisticated techniques. Future research, related to this work, will focus on the application of GLS in practical systems for frequency assignment. The real world is the ultimate testbed for evaluating an algorithm given the many difficulties involved such as side constraints, complex objective functions and user preferences, and also requirements for dynamic or distributed optimisation.

REFERENCES

- [1] Boyce, J.F, Dimitropoulos, C.H.D., vom Scheidt, G., Taylor, J.G., "GENET and Tabu Search for combinatorial optimisation problems", *Proceedings of World Congress on Neural Networks*, Washington D.C., INNS Press, 1995.
- [2] Bentley, J.L., "Fast Algorithms for Geometric Traveling Salesman Problems", *ORSA Journal on Computing*, Vol. 4, 387-411, 1992.
- [3] Benthem, H., Hipolito, A., Jansen, B., Roos, C., Terlaky, T., and Warners, J., "GRAPH: a test case generator, generating radiolink frequency assignment problems heuristically", *EUCLID CALMA technical report*, EUCLID CALMA Project, Delft and Eindhoven Universities of Technology, The Netherlands, 1995.
- [4] Castelino, D.J., Hurley, S., and Stephens, N.M., "A Tabu search algorithm for frequency assignment", *Annals of Operations Research*, Vol 41, pp. 343-358, 1993.
- [5] Castelino, D., and Stephens, N., "Tabu thresholding for the frequency assignment problem", *Meta-Heuristics: Theory and Applications* (Edited by I.H. Osman and J.P. Kelly), Kluwer Academic Publishers 1996.
- [6] Davenport, A., "GENET: a connectionist architecture for constraint satisfaction", *PhD Thesis*, Department of Computer Science, University of Essex, 1997.
- [7] Eindhoven RLFAP Group, "EUCLID CALMA, Radio Link Frequency Assignment Project, Technical Annex-2.3.3: Local search", *EUCLID CALMA technical report*, EUCLID CALMA Project, Eindhoven University of Technology, The Netherlands, 1995.
- [8] Glover, F., and Laguna, M., *Tabu Search*, Kluwer Academic Publishers, Boston, 1997.
- [9] Hale, W.K., "Frequency Assignment: Theory and Applications", *Proceedings of the IEEE*, 68, 1497-1514, 1980.
- [10] Hao J.-K., Dorne, R., Galinier, P., "Tabu Search for Frequency Assignment in Mobile Radio Networks", *Journal of Heuristics*, Volume 4, Issue 1, pp. 47-62, June 1998.
- [11] Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., "Optimisation by Simulated Annealing", *Science*, Vol. 220, 671-680, 1983.
- [12] Kilby, P., Prosser, P., and Shaw, P., "Guided local search for the vehicle routing problem", *Proceedings of the 2nd International Conference on Metaheuristics*, July 1997.
- [13] Minton S., Johnston, M.D., Philips A.B., and Laird, P., "Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems". *Artificial Intelligence*, 58, 161-205, 1992.
- [14] Murphey, R.A., Pardalos, P.M., and Resende, M.G.C., "Frequency Assignment Problems", *AT&T Labs Research Technical Report: 98.16.1*, (to appear in *Handbook of Combinatorial Optimisation*, Kluwer Academic Publishers, 1999).
- [15] Reeves, C.R., "Genetic Algorithms", in: Reeves, C. (ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publishing, 151-196, 1993.
- [16] vom Scheid, G., "Extension of GENET to Partial Constraint Satisfaction Problems and Constraint Satisfaction Optimisation Problems", *Master's Thesis*, Dept. of Mathematics, King's College London, UK, 1995.
- [17] Selman, B., Levesque, H. J., and Mitchell, D.G., "A new method for solving hard satisfiability problems", *Proceedings of AAAI-92*, 440-446, 1992.
- [18] Tiourine, S., Hurkins, C., and Lenstra, J.K., "An overview of algorithmic approaches to frequency assignment problems", *EUCLID CALMA Project Overview Report*, Delft University of Technology, The Netherlands, 1995.
- [19] Tsang, E, *Foundations of Constraint Satisfaction*. Academic Press, 1993.
- [20] Tsang, E., and Voudouris, C., "Fast local search and guided local search and their application to British Telecom's workforce scheduling problem", *Operations Research Letters*, Vol. 20, No. 3, 119-127, 1997.
- [21] Voudouris, C., *Guided Local Search for Combinatorial Optimization Problems, PhD Thesis*, Department of Computer Science, University of Essex, Colchester, UK, July, 1997.
- [22] Voudouris, C., and Tsang, E., "Partial Constraint Satisfaction Problems and Guided Local Search", *Proceedings of 2nd Int. Conf. on Practical Application of Constraint Technology (PACT'96)*, London, April, 337-356, 1996.
- [23] Voudouris, C., and Tsang, E., "Guided local search and its application to the travelling salesman problem", *European Journal of Operational Research*, Vol. 132, Issue 2, pp80-110, 1998.
- [24] Voudouris, C., and Tsang, E., "Guided Local Search for the Radio Link Frequency Assignment Problem", *manuscript to be submitted to the Journal of Heuristics*.
- [25] Wang, C.J., and Tsang, E., "Solving constraint satisfaction problems using neural-networks", *Proceedings of IEE Second International Conference on Artificial Neural Networks*, 295-299, 1991.

CELAR Set	Soluble Instances						Insoluble Instances						
Method	1	2	3	4	5	11	Time	6	7	8	9	10	Time
Simulated Annealing (EUT)	2	0	2	0	0	2	1min	6%	65%	5%	0%	0%	310min
Taboo Search (EUT)	2	0	2	0	-	0	5min	-	-	-	-	-	-
Variable Depth Search (EUT)	2	0	2	0	-	10	6min	3%	0%	14%	0%	0%	85min
Simulated annealing (CERT)	4	0	0	0	-	10	41min	42%	1299%	70%	2%	0%	42min
Tabu Search (KCL)	2	0	0	0	0	2	40min	167%	1804%	566%	8%	1%	111min
Extended GENET (KCL)	0	0	0	0	0	2	2min	12%	27%	40%	-	-	20min
Genetic Algorithms (UEA)	6	0	2	0	-	10	24min	0%	386%	134%	3%	0%	120min
Genetic Algorithms (LU)	-	-	-	-	-	-	-	0%	0%	0%	0%	0%	hours
GLS-FLS-S4	0	0	0	0	0	2	2.91min	-1.34%	0.03%	0%	0%	0%	12min
CALMA Project Best Solution	16*	14*	14*	46*	792*	22*		3437	343594	262	15571	31516	

Table 3 Comparison of the GLS with the heuristic search algorithms of the CALMA project on the Celar Instances.

GRAPH Set	Soluble Instances								Insoluble Instances							
Method	1	2	3	4	8	9	10	14	Time	5	6	7	11	12	13	Time
Simulated Annealing (EUT)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Taboo Search (EUT)	0	0	0	0	0	0	398	0	3min	-	-	-	-	-	-	-
Variable Depth Search (EUT)	-	-	-	-	-	-	-	-	-	1%	0%	0%	0%	0%	0%	63min
Simulated annealing (CERT)	4	4	-	-	-	-	-	-	22min	-	123%	23%	-	-	-	81min
Tabu Search (KCL)	0	2	-	-	4	4	-	2	28min	409%	18%	303%	2648%	186%	1727%	minutes
Extended GENET (KCL)	0	0	-	-	2	4	-	-	5min	-	-	-	-	-	-	-
Genetic Algorithms (UEA)	2	2	i	i	-	10	i	4	9min	33%	282%	39%	763%	29%	342%	20min
Genetic Algorithms (LU)	-	-	-	-	-	-	-	-	-	0%	3%	0%	-	0%	104%	hours
GLS-FLS-S4	0	0	0	0	0	0	0	-2	1.29min	0%	-1.17%	0.14%	-12.12%	0%	-6.87%	5.46min
CALMA Project Best Solution	18*	14*	380*	394*	20	18*	394	10		221	4189	4324	3513	11827	11130	

Table 4 Comparison of the GLS with the heuristic search algorithms of the CALMA project on the Graph Instances.

soluble instances

4 Number of frequencies above optimal solution

i infeasible solution reported

insoluble instances

42% deviation from the best known solution

- method not applicable or no solution is reported

* optimality of solution is proved

+ preprocessing time not included

Research Centers:

CERT - Centre d'Etudes et de Recherches de Toulouse, France;

DUT - Delft University of Technology, The Netherlands;

EUT - Eindhoven University of Technology, The Netherlands;

KCL - King's College London, United Kingdom;

LU - Limburg University, Maastricht, The Netherlands;

UEA - University of East Anglia, Norwich, United Kingdom.

Machines Used:

CERT - SUN SPARC 10

DUT - HP9000/720

EUT - SUN SPARC 4

KCL - DEC Alpha 3000 (130 Mhz)

LU (GA) - DEC OSF/1 AXP

LU(CS) - PC

UEA - DEC Alpha (133 Mhz)

GLS - UltraSparc (167 Mhz)