CHANNEL ASSIGNMENT IN CELLULAR COMMUNICATION USING A GREAT DELUGE HYPER-HEURISTIC

Graham Kendall and Mazlan Mohamad
Automated Scheduling, Optimisation and Planning (ASAP) Research Group,
University of Nottingham, School of Computer Science and IT
Nottingham NG8 1BB, UK

Abstract - This paper proposes a methodology for the channel assignment problem in the cellular communication industry. The problem considers the assignment of a limited channel bandwidth to satisfy a growing channel demand without violating electromagnetic interference constraints. The initial solution is generated using random constructive heuristic. This solution is then improved using a hyper-heuristic technique based on the great deluge algorithm. Our experimental results, on benchmarks data sets, gives promising results.

1. INTRODUCTION

The choice of a cellular network as a communication platform has significantly increased in recent years and will continue to do so as mobile phones become more widespread and the number of services available increases. This phenomenon results in a requirement for efficiently allocate a limited frequency bandwidth to create extra channel capacity and coverage in a cellular network. The definition of channel refers to the access methods used in a cellular system. If the cellular system uses frequency division multiple access (FDMA), the channel is referred to as a frequency slot. It is referred to as a time slot in time division multiple access (TDMA) system. The 2nd generation cellular system, Global System for Mobile Communications (GSM) uses a combination of both FDMA/TDMA.

The methods available to increase channel capacity and coverage in a cellular network comprise of frequency reuse and cell splitting. Frequency reuse involves using the same frequency or channel simultaneously in other cells subject to the base transceivers station (BTS) distance. Cell splitting splits a larger cell into more than one cell to cover a particular geographical area. Each cell covers a smaller area, with a lower transmission power and thus offers the ability to reuse frequencies more often.

The problem with these methods is the electromagnetic interference between channels in the same cell (co-site channel constraint), interference between neighbouring cells (adjacent channel constraint) and interference between other cells utilising the same channel (co-channel constraint).

For a co-site channel constraint, the channels in the same site need to be separated by a minimum distance in the frequency domain. For an adjacent channel constraint, adjacent or neighborhood cells cannot use an adjacent channel simultaneously. For a co-channel constraint, the same channel cannot be assigned to certain pairs of radio cells simultaneously.

The offline task of allocating a set of radio channels to meet the requested traffic demand for a given numbers of calls is referred to as the fixed channel assignment problem (CAP) or the fixed frequency assignment problem (FAP). The term ‘fixed’ refers to the fact that channels are permanently assigned to a particular cell. The variant of this fixed scheme is dynamic CAP, where all channels are located in a central pool and dynamically assigned based on channel requests. In the 1st and 2nd generation cellular system, the performance of fixed CAP outperformed dynamic CAP under heavy traffic loads and uniform traffic distribution [1].

There are two types of task involved in fixed CAP i.e. [2]:

1. Minimum span problem (MS-CAP)

Given a traffic demand, cell station number and compatibility matrix, find the minimum number of consecutive channels used with free electromagnetic interference i.e.

Minimise the number of radio channels
s.t. traffic demand and interference constraint

2. Minimum interference problem (MI-CAP)

Given a fixed number of radio channels, cell station number, traffic demand and compatibility matrix, minimise severity of channel interference i.e.

Minimise severity of channel interferences
s.t. demand constraints

In this paper, we only consider the minimum span problem (MS-CAP).This paper is organized as follows. In Section 2 we discuss the problem description and mathematical representation. In Section 3 and 4 we then describe our proposed methodology followed by experimental result and conclusion in section 5 and 6.

2. MS-CAP PROBLEM DESCRIPTION

In the 1st and 2nd generation of cellular systems, the frequency spectrum is divided into evenly spaced consecutive channels using frequency division (FD) or time division (TD) [3]. Therefore, the channels are represented by positive integers 1, 2, 3,…,Z where Z is a maximum number of available channels. The basic model of MS-CAP can be represented as follows (mostly adopted from [4,5])

- N : a set of cells in the network, where $N = (cell_1, cell_2, .., cell_N)$.
- D : demand vector $D = (d_1, d_2, ..., d_N)$ where d_i is the number of radio channels required in $cell_i$ in order to satisfy the channel demand.
- C : compatibility matrix, C_{NxN}, where each element C_{ij} denotes the frequency separation required between $cell_i$ and $cell_j$.
- f_{ik} : k^{th} channel assigned to $cell_i$.

Therefore, the objective of MS-CAP is to [6]

Minimise Z

Subject to

$\sum_{k=1}^{Z} f_{ik} = d_i$ for $1 \le i \le N$

$|k - l| \ge C_{ij}$ for $1 \le k,l \le Z$ and $1 \le i,j \le N$ such that $f_{ik} = f_{jl} = 1$

$f_{ik} = 0$ if channel k is not assigned to $cell_i$, otherwise 1, for $1 \le k,l \le Z$ and $1 \le i,j \le N$

3. A GREAT DELUGE HYPER-HEURISTIC FOR MS-CAP

Many heuristics are problem dependent, meaning that one heuristic cannot be used to solve different problems [1]. In order to alleviate this problem, the concept of a hyper-heuristic was introduced, which is, a (meta-)heuristics that operates on (meta-)heuristics [7]. Hyper-heuristics are problem independent and have been successfully applied to various optimisation and scheduling problem [8,9,10,11].

Hyper-heuristics operate at a higher level of abstraction than meta-heuristics, managing a set of low level heuristics (LLH) without knowledge of the problem domain. The general framework can be shown in figure 1[7].

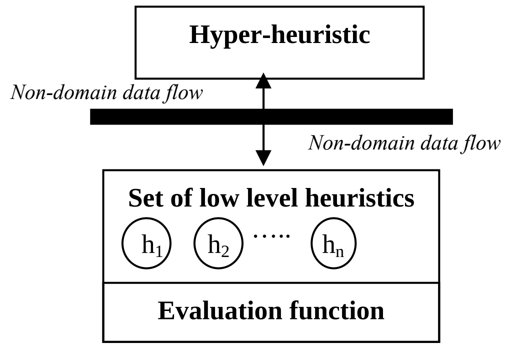

Figure 1: Hyper-heuristic Framework

There is a barrier between the LLHs and hyper-heuristic, meaning that only non-domain data can cross the barrier. Referring to the MS-CAP case, only the quality of solution (value of Z) and computation time is allowed to cross the domain barrier. The hyper-heuristic has no knowledge of the problem it is trying to solve, only that it has a set of LLHs, which it can call, and whether it is trying to minimise or maximise the evaluation function. The set of LLHs act as simple local search or other problem specific heuristics. Each LLH will typically, search a different neighborhood structure and each move by the LLHs will produce a new solution. The hyper-heuristic will make a decision as to whether to accept or reject the new solution. The hyper-heuristic is also used to decide as to which LLH to call at each iteration.

By continuously applying a single LLH, it can lead the final solution trapped in a local optima. One diversification strategy is to change the neighbourhood structure, i.e. by calling different LLHs. This is similar to the idea of Variable Neighbourhood Search(VNS) [12]. However, whereas VNS keeps applying a single heuristic, until it gets trapped in a local optima, hyper-heuristics have a 'free' choice at each decision point which heuristic to call. Previous works on hyper-heuristic have mainly concentrated on the calling sequence of LLHs. The choice function hyper-heuristic [13] ranks the solution quality performance of each LLH. Based on the previous quality result; the hyper-heuristic will decide which LLH should be called next. A genetic algorithm hyper-heuristic [9] used a allele in a chromosome to represent each LLH. The quality of the solution is evaluated based on the sequence of calls within the chromosome. Another approach has used a tabu list to monitor the performance of each LLH in solving an examination timetabling problem [11]. Instead of concentrating on LLH sequences, a Monte Carlo hyper-heuristic [10] used an acceptance criteria of solution quality performed by randomly calling LLHs.

We have used the hyper-heuristic method from [10] but propose a different acceptance criteria as used by G. Dueck [14]. The method, namely the *Great Deluge Algorithm (GDA),* was extended and the performance of the algorithm as a local search meta-heuristic was evaluated and gave a superior result on examination timetabling problem [15].

We can model MS-CAP as local search framework by defining a solution space S, an objective function f and a neighborhood N. The basic concept of the *Great Deluge Hyper-heuristic Algorithm* can be shown by the pseudocode in figure 2:

```
Step 1: Initialisation
    (a) Choose an initial solution $s_o \in S$
    (b) Compute an initial objective function $f(s_o)$
    (c) Set Initial LEVEL = $f(s_o)$
    (d) Set DownRate value
Step 2 : Operation and termination
    (a) Call LLH to generate new neighbour solution $s_n \in N(s_o)$
    (b) Compute neighbour objective function $f(s_n)$
```

```
(c) if $f(s_n) \leq LEVEL$, accept $f(s_n)$ then update $s_o = s_n$
(d) Reduce $LEVEL = LEVEL - DownRate$
(e) If stopping condition = false, go to step 2(a)
```

Figure 2: Great Deluge Hyper-heuristic Algorithm

During initialisation stage, the objective function $f(s_o)$ (which is Z value) is set as *LEVEL* at the beginning of iteration, it is slowly reduce with *DownRate* value at every iterations. We can say is that, the performance of this algorithm is dependent on the choice on *single parameter (DownRate)* and the starting value of *LEVEL*. In our case, we define *DownRate* using the concept adopted from [15] and shown in figure 3:

$$DownRate = (f(s_o) - LB) / Iter$$

Where

LB = Best result

Iter = Number of iteration

Figure 3: Calculation of *DownRate*

Here, *LB* we define as best result found in the literature [16,17] and the number of iteration, *Iter* based on the suggestion in [4], which is $Iter = N(N-1)/2$. Because of the algorithm can easily hit the *LB* for easier benchmarks problem, we set our stooping criteria is define based on 'hit the *LB*' as priority stopping condition. For difficult benchmarks problem, we used run time duration as our stooping criteria, which is we set as 480 seconds for the whole run.

Based on our preliminary experiment, the calling sequence of LLHs did not contribute or affect much in solution quality. Therefore, in this paper, we only used randomly called LLHs, but it can be extended to use other methods of calling LLHs as in [8,9,13]

The proposed method is similar to a random decent method, which only accept an improve solution, but in our proposed algorithm, the acceptance of the new solution will be decreased according to *DownRate* value. As long as the returned solution is below the current value of *LEVEL*, the hyper-heuristic will accept this solution.

4. Low level heuristics

We have created four simple 2- opt local search methods to act as our low level heuristics (LLHs). The definition of each LLHs is as follows:

- h_1 - Sort the channel from lowest to highest, delete the call with the highest channel assignment, randomly insert at any point and reassign the channel.
- h_2 - Same as h_1, but randomly select the call to delete.
- h_3 - Same as h_1, but find the best point at which to reassign.
- h_4 - Same as h_1, but randomly change the call order starting from insertion point.

All the proposed LLHs will create unique moves with zero violating constraint with faster computation time except for h_3, which act like a steepest decent heuristic. h_3 will find the best neighbour in the current nighbourhood structure; therefore it needs more computation time.

5. Testing and results

We have implemented and tested our algorithm on a Pentium III-700 MHz computer. We compare our performance with [16], which proposed an algorithm that generates a population of random valid solutions using a quadnary representation [0,+1,-1,+9], which means [assignable,used,unassignable,unused].

Another comparison is with the latest work of Ghosh *et al.* [17], where they use a genetic algorithm based on geometric symmetry and Batiti *et. al* [18], where they used a combination of randomised saturation degree and local search approach. We implemented three different networks size (21, 25 and 55) with different compatibility matrix, C, and traffic demand, D. The details C and D can be found in [16].

We use a random constructive heuristic to generate an initial solution and use two stopping criteria (reach the best known result or the runtime is expired). We ran each experiment 10 times. By applying our approach we were able to achieve promising results as presented in table 1 and table 2 which respectively shown the average of minimum bandwidth (Z) and time taken to produce the result.

The results show that we found the lower bound on 11 out of 20 benchmarks problem with reasonable computation times. For the more difficult problems, (e.g. test 11 and test 12) with the maximum allowed time (480 seconds) and 'poor' initial solutions, we still manage to improve the solution quality by 26.4% and 16.3% respectively. As stated in [17], these are the most difficult benchmarks problems and they had run times of 16-80 hours in order to achieve their solutions. Also, due to the random element in our LLHs, the computation time varies for each run. For example, the computation time for test 1 varies from 17s to 102s and for test 3, the computation time varies from 4s to 111s.

6. Conclusion

In this work, we have proposed an algorithm based on a hyper-heuristic approach. As a problem independent, methodology hyper-heuristics are a rapid development tool for optimisation problems. We use a greedy constructive heuristic to generate an initial solution then apply LLHs to improve the solution quality. Results shows that our proposed algorithm can achieve promising results for all benchmarks problem, even though we can not produce better quality solutions compared to previous work for some of the benchmark problems. However, our results are competitive, if not superior with respect to run time. In future work, we will use other acceptance criteria to compare against our proposed great deluge acceptance criteria and, we will increase the number of low level heuristics. We will also use different strategies to produce initial solutions, such as the using of randomised saturation degree (RSD), which give superior result in previous work [18], whereby they used the combination of RSD for 'good' initial solution and enhanced those 'unable to get good solution' benchmarks with adaptive local search approach. With a good initial solution, intelligent LLHs and enhanced our approach by using the combination of calling sequence and acceptance criteria, we hope to produce even better results in the future.

TABLE 1 : PERFORMANCE COMPARISON

Test	C_Matric(C)/ Demand(D)	Trivial Lower Bound	Chakraborty [16]	Ghosh *et al.*[17]	Batiti *et. al* [18]	Our Approach Initial/average
1	C1_21/D1_21	533	533	533	533	**538/533**
2	C1_21/D2_21	309	309	309	309	**312/309**
3	C2_21/D1_21	533	533	533	533	**540/533**
4	C2_21/D2_21	309	309	309	309	**371/309**
5	C3_21/D1_21	457	457	-	-	**488/457**
6	C3_21/D2_21	265	265	-	-	**268/265**
7	C4_21/D1_21	457	457	-	-	**634/457**
8	C4_21/D2_21	265	280		-	**350/274**
9	C5_21/D1_21	381	381	381	381	**433/381**
10	C5_21/D2_21	221	221	221	221	**235/221**
11	C6_21/D1_21	381	463	427	427	598/440
12	C6_21/D2_21	221	273	253	254	318/266
13	C7_21/D1_21	305	305	-	-	**396/305**
14	C7_21/D2_21	177	197	-	-	**222/187**
15	C8_21/D1_21	305	465	-	-	**538/451**
16	C8_21/D2_21	177	278	-	-	**323/275**
17	C1_25/D3_25	21	73	-	73	**78/73**
18	C1_25/D4_25	89	121	-	-	206/200
19	C1_55/D5_55	309	309	-	-	**315/309**
20	C1_55/D6_55	71	79	-	-	**99/71**

TABLE 2 : COMPUTATION TIME COMPARISON

Test	C_Matric(C)/ Demand(D)	Chakraborty[16] *(DEC Alpha station)*	Ghosh *et al.* [17]*(DEC Alpha Station)*	Batiti *et. al* [18] *(DEC AlphaServer)*	Our Approach (average time)
1	C1_21/D1_21	8.2s	0.5-1s	<1s	40s
2	C1_21/D2_21	6.0s	0.5-1s	<1s	0.8s
3	C2_21/D1_21	11.1s	6-12s	<1s	46s
4	C2_21/D2_21	10.2s	6-17	<1s	29s
5	C3_21/D1_21	8.9s	-	-	81s
6	C3_21/D2_21	8.1s	-	-	14s
7	C4_21/D1_21	9.8s	-	-	300s
8	C4_21/D2_21	7.9s	-	-	480s
9	C5_21/D1_21	7.5s	2-5s	<1s	38s
10	C5_21/D2_21	6.9s	2-7s	<1ss	25s
11	C6_21/D1_21	9.5s	-	<30s	480s
12	C6_21/D2_21	7.7s	-	<35s	480s

Test	C_Matric(C)/ Demand(D)	Chakraborty[16] *(DEC Alpha station)*	Ghosh *et al.* [17]*(DEC Alpha Station)*	Batiti *et. al* [18] *(DEC AlphaServer)*	Our Approach (average time)
13	C7_21/D1_21	7.3s	-	-	147s
14	C7_21/D2_21	6.8s	-	-	480s
15	C8_21/D1_21	8.4s	-	-	480s
16	C8_21/D2_21	7.5s	-	-	480s
17	C1_25/D3_25	1.9s	-	<1s	480s
18	C1_25/D4_25	6.3s	-	-	480s
19	C1_55/D5_55	24.5s	-	-	120s
20	C1_55/D6_55	16.7s	-	-	75s

7. References

[1] H.G. Sandalidis and P. Stavroulakis, Heuristics for Solving Fixed-Channel Assignment Problems, ch. 3. In: I. Stojmenovic (ed) *Handbook of Wireless Network and Mobile Computing*, John Wiley & Sons, pp. 51-70,2002.

[2] K. Smith and M. Palaniswami, Static and Dynamic Channel Assignment Using Neural Networks, *IEEE Journal On Selected Areas In Communication*, Vol. 15, No.2, pp. 238-249, February 1997.

[3] I. Katzela and M. Naghshineh, Channel Assignment Schemes for Cellular Mobile Telecommunication Systems - A Comprehensive Survey, *IEEE Personal Communications Magazine*, June 1996.

[4] W. Wang and C.K. Rushforth, An Adaptive Local-Search Algorithm for the Channel-Assignment Problem (CAP), *IEEE Trans. Veh. Technology*, vol 45, pp. 459-446, August 1996.

[5] K.N. Sivarajan, R.J. McEliece and J.W. Ketchum, Channel assignment in cellular radio, in Proc. 39th *IEEE Veh. Technology. Soc. Conf.,* pp. 846-850, May 1989.

[6] B. Dirk and K. Ulrich, A New Strategy for Application of Genetic Algorithms to the Channel-Assignment Problem. *IEEE Trans. Veh. Technology,* vol 48, no. 4, pp. 1261-1269, July 1999.

[7] E. Burke, E. Hart, G. Kendall, J. Newall, P. Ross and S. Schulenburg, Hyper-Heuristics: An emerging direction in modern search technology, ch. 16. In: F. Glover, and G. Kochenberger, (ed) *Handbook of Meta-Heuristics* , pp 457-474, Kluwer, 2003.

[8] E.K Burke, G Kendall and E Soubeiga, A Tabu-Search Hyper-Heuristic for Timetabling and Rostering. *Journal of Heuristics*, 9(6), pp. 451-470, 2003.

[9] L. Han and G. Kendall, Investigation of a Tabu Assisted Hyper-Heuristic Genetic Algorithm. *In proceedings of Congress on Evolutionary Computation(CEC2003)*, vol. 3, pp. 2230-2237, Canberra, Australia.

[10] M. Ayob and G. Kendall, A Monte Carlo Hyper-Heuristic To Optimise Component Placement Sequencing For Multi head Placement Machine, *InTech'03* Thailand, pp. 132-141, ISBN 974-658-151-1.

[11] G. Kendall and N. Mohd Hussin , An Investigation of a Tabu Search Based Hyper-heuristic for Examination Timetabling, Accepted for publication in *selected papers from MISTA2003*, Kluwer Publication eds G.Kendall, E.Burke and S. Petrovic.

[12] N. Mladenovic and P. Hansen, Variable neighborhood search, *Computer in Operations Research*, vol. 24, pp. 1097-1100, 1997.

[13] E. Soubeiga, Development and Application of Hyperheuristics to Personnel Scheduling, *PhD Thesis*, Department of Computer Science, University of Nottingham, UK, June 2003.

[14] G.Dueck, New Optimization Hueristics : The Great Deluge Algorithm and the Record-to Record Travel, *Journal of Computational Physics*, 104, pp. 86-92, 1993.

[15] Y. Bykov, Time-Predefined and Trajectory-Based Search: Single and Multiobjective Approaches to Exam Timetabling, *PhD Thesis*, Department of Computer Science, University of Nottingham, UK, November 2003.

[16] G.Chakraborty, An Efficient Heuristic Algorithm for Channel Assignment Problem in Cellular Radio Networks, *IEEE Trans. Veh. Technology*, vol. 50, no. 6, pp. 1528-1539, Nov 2001.

[17] S.C. Ghosh, B.P. Sinha and N. Das, Channel Assignment Using Genetic Algorithm Based on Geometric Symmetry, *IEEE Trans. Veh. Technology*, vol. 52, no. 4, pp. 860-875, July 2003.

[18] R.Batiti, A.Bertossi and D.Cavallaro, A Randomized Saturation Degree Heuristic for Channel Assignment in Cellular Radio Networks, *IEEE Trans on Veh Technology,* vol 50, No.2, pp. 364-374, March 2001.