Optimization of the Assignment of Base Stations to Base Station Controllers in GERAN

M. Toril, V. Wille

Abstract—In GSM-EDGE Radio Access Network (GERAN), the assignment of base stations to base station controllers has a strong impact on network performance. In this paper, an exact method is proposed to improve an existing assignment of base stations to base station controllers. The aim is to minimize the number of handovers between base station controllers given a maximum number of base stations re-allocations. Results from a real case study show that the proposed method outperforms classical re-partitioning approaches in terms of solution quality.

Index Terms—Mobile communication, Optimization methods, Load management.

I. INTRODUCTION

In cellular networks, groups of Base Stations (BSs) are governed by a single Base Station Controller (BSC) to ensure scalability. In GERAN, the assignment of BSs to BSCs has a strong impact on handover performance. Thus, handovers between BSs connected to different BSCs consume much more signaling resources and experience longer delays than those between BSs in the same BSC. Therefore, adjacent BSs should be connected to the same BSC, provided that the BSC capacity limit is not exceeded. During network design, network planners choose the cluster of BSs assigned to each BSC so that the number of handovers between BSCs is minimized. During network operation, the inclusion of new BSs often forces a re-configuration of the network to equalize the load across BSCs. In the past, this problem has been solved manually, leading to sub-optimal network performance despite the time and effort invested. Hence, operators demand automatic tools to solve the problem network wide and periodically.

Several authors have formulated the assignment of BSs to BSCs as a graph partitioning problem [1], which is solved by exact [2] or heuristic [3][4][5] methods. However, all proposed methods follow a green field approach, where the solutions thus obtained lead to sub-optimal network performance due to the constraints and adjacency between BSs. In the past, this problem has been solved manually, leading to sub-optimal network performance despite the time and effort invested. Hence, operators demand automatic tools to solve the problem network wide and periodically.

Several authors have formulated the assignment of BSs to BSCs as a graph partitioning problem [1], which is solved by exact [2] or heuristic [3][4][5] methods. However, all proposed methods follow a green field approach, where the solutions thus obtained lead to sub-optimal network performance due to the constraints and adjacency between BSs. In the past, this problem has been solved manually, leading to sub-optimal network performance despite the time and effort invested. Hence, operators demand automatic tools to solve the problem network wide and periodically.

II. PROBLEM FORMULATION

In a live network, all BSs in a site must be assigned to the same BSC to share part of the network infrastructure. Thus, it is sites and not BSs that are assigned to BSCs. To find a proper assignment, the network is modeled by a graph, whose vertices, \( V \), and edges, \( E \), represent the sites and adjacencies between sites, respectively. The weight of vertex \( i \), \( w_i \), is the carried traffic in site \( i \), while the weight of edge \( (i,j) \), \( \gamma_{ij} \), is the number of handovers between sites \( i \) and \( j \). The partitioning of the graph, performed by assigning vertices to a set of subdomains, \( N \), reflects the assignment of sites to BSCs. Any partition defines a set of edges joining vertices in different subdomains, whose sum of weights represents the number of handovers between sites in different BSCs. Thus, minimizing the latter quantity is a graph partitioning problem that can be formulated as the Integer Linear Programming (ILP) model:

\[
\begin{align*}
\text{Min} \quad & \sum_{(i,j) \in E} \gamma_{ij}(1 - \sum_{n \in N} Z_{ijn}) & (1) \\
\text{s.t.} \quad & \sum_{n \in N} X_{in} = 1, & \forall i \in V, (2) \\
& \sum_{i \in V} w_i X_{in} \leq B_{aw}, & \forall n \in N, (3) \\
& \sum_{i \in V} \omega_i (X_{im} - B_{ru} X_{in}) \leq 0, & \forall m, n \in N, (4) \\
& \sum_{i \in V} \sum_{n \in N, n \neq X_{n}(i)} X_{in} \leq B_{ch}, & (5) \\
& Z_{ijn} \leq X_{in}, & \forall (i,j) \in E, n \in N, (6) \\
& Z_{ijn} \leq X_{jn}, & \forall (i,j) \in E, n \in N, (7) \\
& Z_{ijn} \geq X_{in} + X_{jn} - 1, & \forall (i,j) \in E, n \in N, (8) \\
& X_{in} \in \{0, 1\}, & \forall i \in V, n \in N, (9) \\
& Z_{ijn} \in \{0, 1\}, & \forall (i,j) \in E, n \in N, (10)
\end{align*}
\]

where \( X_{in} \) and \( Z_{ijn} \) are binary variables that reflect the assignment of site \( i \) and adjacency \( (i,j) \) to subdomain \( n \), respectively.
respectively, and \( X_0(i) \) is the BSC to which site \( i \) is assigned in the solution currently implemented in the network. (1) reflects the goal of minimizing the number of handovers between sites in different BSCs. (2) ensures that a site belongs to only one BSC. (3) reflects the BSC capacity limit, \( B_{aw} \), while (4) ensures that the traffic is evenly distributed among BSCs by limiting the load imbalance ratio between any pair of BSCs to \( B_{rw} \). (5) restricts the number of changes with respect to the current solution to \( B_{ch} \). (6)-(8) show the dependence between decision variables by linear constraints and (9)-(10) are binary constraints.

To discard non-feasible site-BSC pairs due to geographical constraints, a proximity matrix, \( A = [a_{in}] \), is defined, where \( a_{in} \) equals to 1 if site \( i \) can be connected to BSC \( n \), 0 otherwise. For simplicity, it is considered here that a site can be connected to a BSC if the site-BSC distance is below a certain threshold, \( B_d \). Unlike in [2], cabling costs are not included in the objective function. It is assumed that the main penalty of re-allocating a site is due to maintenance actions in the source and target BSCs and the transmission network. If this is not the case, a cabling cost term could easily be included in (1). The previous data is used to fix variables and eliminate constraints in the model. Thus, \( X_{in} = 0 \) if \( a_{in} = 0 \), \( X_{jn} = 1 \) if \( a_{jn} = 1 \) and \( a_{im} = 0 \) \( \forall \ m \neq n \). Likewise, \( Z_{ijn} = 0 \) if \( X_{in} = 0 \) or \( X_{jn} = 0 \), and \( Z_{ijn} = 1 \) if \( X_{in} = X_{jn} = 1 \).

Constraint (5), not considered in [2], has a strong impact on algorithmic performance. As a constraint, it reduces the number of feasible solutions, thus limiting the benefit of optimization. However, the efficiency of enumerative algorithms is improved as the number of solutions to be evaluated is significantly less. This makes the exact approach viable.

III. SOLUTION METHOD

Network operators need to evaluate the benefit of every new re-assignment, as the maximum number of changes is not fixed a priori. Hence, a series of problems must be solved to find the best solution to the model in (1)-(10) for different values of \( B_{ch} \). To solve each model, the Branch-and-Cut (BC) algorithm [7] is used. This algorithm combines the Branch-and-Bound (BB) and Cutting Planes (CP) algorithms. The BC algorithm is a refined enumeration method that discards groups of non-promising solutions without explicitly testing them. The algorithm starts by computing lower and upper bounds for the optimal value of the original problem. If both bounds coincide and are feasible, an optimal solution has been found and the procedure terminates. Otherwise, the original problem is divided into several problems (referred to as \textit{branching}) through the addition of constraints. These constraints divide the complete solution space into complementary regions. The algorithm is applied recursively, generating a tree of subproblems. If an optimal feasible solution is found to a subproblem, then it is feasible to the original problem, but it is not necessarily the globally optimal solution. These solutions are used to prune branches whose lower bound is higher than the best value found so far. The search continues until there are no unexplored parts of the solution space. The CP algorithm is used to tighten bounds by adding constraints to the problem (referred to as \textit{cuts}) without affecting the feasible region. The reader is referred to [7] for more details on the BC algorithm.

To speed up the search, a heuristic solution is fed to the BC algorithm to obtain an upper bound for the problem. In this work, ILP models are solved in increasing order of \( B_{ch} \) to exploit that the optimal solution for \( B_{ch} = c \) is necessarily feasible for \( B_{ch} = c + 1 \). Thus, the optimal value for \( B_{ch} = c \) is an upper bound of the optimal value for \( B_{ch} = c + 1 \).

IV. PERFORMANCE ANALYSIS

A. Analysis Methodology

The above-described method has been tested in a real problem instance. The dataset corresponds to a network area comprising 319 sites distributed over 6 BSCs in a GSM network. The network model is built from handover and traffic measurements for one week. As optimization constraints, the maximum traffic per BSC, the maximum load imbalance ratio and the maximum site-BSC distance are set to the values in the existing solution (i.e., \( B_{aw} = 11076 \text{E}, B_{rw} = 1.12 \) and \( B_{ch} = 57 \text{km} \)).

Three methods are compared. The first two are classical partitioning heuristics based on local refinement algorithms. The \textit{Greedy Refinement} (GR) [1] is the approach followed by most operators. This algorithm consists of a series of iterations, where all possible re-assignments are evaluated and the best movement is selected in a greedy fashion. The solution is updated and the refinement process proceeds until no change improves the previous solution. Experience shows that this method tends to get trapped in local minima. To circumvent this limitation, the \textit{Fiduccia-Mattheyses} (FM) algorithm [1] explores moves that temporarily degrade the value of the objective function in the hope that this will lead to better solutions. This algorithm is used for benchmarking in most graph partitioning applications. The third one is the exact method that solves the ILP model of the problem by the BC algorithm (denoted as BC). Heuristic methods are implemented in Matlab, whereas the exact method uses the BC algorithm in GLPK (GNU Linear Programming Kit) [8], available in the public domain. All methods are executed on a 2.4GHz 2GB-RAM Windows-XP computer. For comparison purposes, the initial operator’s solution (denoted as IO) is also included.

The main performance criteria are the total number of handovers between sites in different BSCs and the execution time. These two indicators are evaluated for \( B_{ch} \) ranging from 1 to 10% of the number of sites. Assessment is based on the best solution and the total execution time of the whole series.

B. Analysis Results

Figure 1 compares the number of inter-BSC handovers obtained by the methods for different maximum number of sites re-allocated. The primary axis represents the raw figures, while the secondary axis represents the same numbers normalized to the value of the initial operator’s solution. In the figure, it is observed that all methods improve the existing solution significantly even for small values of \( B_{ch} \). This is a clear indication of the need for optimizing the existing site-to-BSC assignment. GR does not get any improvement after the 4th
purposes. Note that an existing site-to-BSC assignment only has to be modified when a new site is added or a significant change in user mobility trends is detected. Such events tend to occur, at most, on a weekly basis. Likewise, note that GLPK is not the best available solver. Experiments have shown that state-of-the-art commercial solvers (e.g., CPLEX or XPRESS-MP) can reduce runtime by more than one order of magnitude. Nonetheless, GLPK manages to solve all problem instances exactly, even if the number of sites is much larger than in instances that could not be solved in [2]. The reason is the limitation of the number of changes, together with increased capabilities of the solver.

V. CONCLUSIONS

In cellular networks, higher traffic demand and smaller cell size have made network structuring a complex issue. In this paper, an exact method has been proposed to optimize an existing site-to-BSC assignment in GERAN. The method applies a branch-and-cut algorithm to an ILP model of the problem that considers the limitation of the number of sites re-allocated. Results on a real problem instance show that the proposed method reduces the number of inter-BSC handovers in the current solution by 30%, outperforming heuristic re-partitioning approaches. It is worth noting that the same approach can be applied to the assignment of BSs to location areas and mobile switching centers. Likewise, although the method has been applied to GERAN, it is equally valid for other cellular radio access technologies.

REFERENCES


