An Efficient Integer Programming Formulation for the Assignment of Base Stations to Controllers in Cellular Networks

M. Toril [a,*], P. Guerrero-García [b], S. Luna-Ramírez [a], V. Wille [c]

[a] *Communications Engineering Dept., E.T.S.I. Telecomunicación, University of Málaga, Campus Universitario de Teatinos, s/n, E-29010 Málaga, Spain.*
Tel.: +34 952137120, Fax: +34 952132027

[b] *Applied Mathematics Dept., University of Málaga, E-29010 Málaga, Spain*

[c] *Performance Services, Nokia Siemens Networks, Ermine Business Park, Huntingdon, Cambs. PE29 6YJ, UK*

Abstract

In mobile networks, the assignment of base stations to controllers when planning the network has a strong impact on network performance. In a previous paper, the authors formulated the assignment of base stations to packet controllers in GSM-EDGE Radio Access Network (GERAN) as a graph partitioning problem, which was solved by a heuristic method. In this paper, an exact method is presented to find optimal solutions that can be used as a benchmark. The proposed method is based on an effective re-formulation of the classical integer linear programming model of the graph partitioning problem, which is solved by the branch-and-cut algorithm in a commercial optimization package. Performance assessment is based on an extensive set of problem instances built from data of a live network. Preliminary analysis shows some properties of the graphs in this problem justifying the limitations of heuristic approaches and the need for more sophisticated methods. Results show that the proposed method outperforms classical heuristic algorithms used for benchmarking, even under runtime constraints. Likewise, it improves the efficiency of exact methods previously applied to similar problems in the cellular field.

Key words: mobile network optimization, network structuring, integer linear programming, packet control unit, graph partitioning

1. Introduction

Due to their large size and heterogeneity, mobile telecommunications networks have a hierarchical structure for the sake of scalability. In these networks, new elements have to be continuously added to cope with the steady growth of traffic demand. The inclusion of these elements often forces a re-configuration of network hierarchy to balance the load in higher layers. To maximize network performance, given the capacity constraints, it is therefore necessary to find the best clustering of elements in lower layers to be assigned to elements of a higher level.

One of these clustering problems is the assignment of base stations to packet control units (PCUs) in a base station controller (BSC) in GERAN [1]. Each BSC has a certain number of PCUs, responsible for the control of packet-data traffic, to which the base stations (or cells) of the BSC have to be associated. This assignment has a strong impact on the quality of packet-data services. Every change of serving cell experienced by a user of these services (event known as a *cell re-selection*) causes an interruption of the data flow. As not all cells are connected to the same PCU, a cell re-selection might cause the re-allocation of the associated data flow to a different PCU. Field trial results have shown that inter-PCU cell re-selections cause far longer service breaks than intra-PCU cell re-selections [2]. Hence, the PCU plan should minimize the number of inter-PCU re-allocations when a cell re-selection takes place. At the same time, the PCU plan should also ensure that the load of all PCUs is within certain limits.

As already recognized in [3], the *cell-to-PCU assignment problem* (CPAP) can be formulated as a classical *graph partitioning problem* (GPP). As the GPP is known to be $\mathcal{NP}$-complete [4], many heuristic algorithms have been proposed to find good solutions efficiently [5]. Results from the application of several heuristic algorithms to the CPAP are presented in a companion paper [6]. However, in most cases, the quality of these solutions remains unknown, since

* Corresponding author.
Email addresses: mtoril@ic.uma.es (M. Toril), pablito@ctima.uma.es (P. Guerrero-García), sluna@ic.uma.es (S. Luna-Ramírez), volker.wille@nsn.com (V. Wille).

it is normally not possible to find the optimal solution for graphs of practical size in reasonable time. Thus, the analysis is often limited to the comparison between different heuristics, where the solution of the *multi-level refinement* algorithm [7,8] is used as a benchmark. In this paper, it is taken advantage of the fact that, unlike graph partitioning problems in other fields, the size of graphs handled in the CPAP is limited. At the same time, CPAP graphs have several peculiarities that degrade the performance of classical heuristic approaches, which justifies the use of more sophisticated methods.

The advent of increased computing capabilities and powerful optimization tools has boosted the interest in exact graph partitioning algorithms. Thus, several studies have evaluated exact algorithms over random graphs [9] or graphs from VLSI circuit design, compiler design and supercomputing applications [10–12]. However, to the authors' knowledge, no comprehensive study has evaluated these algorithms on graphs derived from the design of a live cellular network. In the mobile network literature, the only attempts were made in [13] and [14], where the problem of assigning cells to switches and location areas was formulated by a simple *integer linear programming* (ILP) model, which was later solved by a *branch-and-cut* algorithm [15]. However, performance assessment was based on a simplistic network model assuming a regular hexagonal cell geometry. Likewise, results from exact graph partitioning algorithms reported in other fields might not be applicable, since CPAP graphs have some important differences (as will be shown later).

In this paper, an exact method is proposed for finding optimal solutions to the CPAP. The proposed method is based on the classical ILP model of the GPP [16], which is re-formulated to reduce model symmetry by combining the approaches suggested in [9] and [12]. The resulting model is solved by the *branch-and-cut* algorithm in a commercial optimization package. As such an approach is classical from the theoretical perspective, the analysis is focused on computational issues. Unlike previous work, performance assessment is based on an extensive set of problem instances constructed with data taken from a real network. To justify the study, a preliminary analysis highlights some unique features of the GPP instances in the CPAP. During the subsequent assessment process, the proposed method is compared with three heuristics widely used for benchmarking purposes: the randomized greedy graph growing partitioning algorithm with greedy refinement [17], the multi-level refinement algorithm [7] and the evolutionary multi-level search [18]. Results will confirm that the proposed approach can be used by network operators to solve the CPAP in live networks.

As pointed out in [6], the problem considered here (i.e., the assignment of PCUs in GERAN) should only be considered as an example of clustering problem in mobile network design. Hence, the main contributions of this work are: a) an analysis of the properties of real graphs from cellular network design, justifying the need for more sophisticated graph partitioning methods and which can be used by other researchers to design more effective clustering algorithms for cellular networks, b) an effective ILP model of the graph partitioning problem, which updates classical models with new techniques to reduce model symmetry, and c) a performance analysis of a state-of-the-art ILP solver in a popular commercial optimization suite, which, unlike previous studies, is done over real problem instances. The presented analysis, models and techniques can easily be extended to many other clustering problems arising in mobile network structuring.

The rest of the paper is organized as follows. Section 2 surveys previous work on related topics. Section 3 presents several ILP formulations of the CPAP. Section 4 outlines the branch-and-cut algorithm used to solve ILP models. Section 5 presents the results of the proposed method over a network model built from data of a live GERAN system. Finally, the main conclusions of the study are summarized in section 6.

2. Related Work

The clustering of elements in the different layers of the network hierarchy is the core of hierarchical structuring of large telecommunication networks. Such a problem can be solved as a graph partitioning problem, for which many algorithms have been conceived in domains as diverse as supercomputing [5], image processing [19], compiler design [10], integrated circuit design [20] and inter-networking [21]. The following paragraphs review graph partitioning methods based on exact and approximation algorithms, paying special attention to those applied to wireless networks.

In the graph partitioning literature, several exact algorithms can be found based on implicit enumeration. In [22], a polynomial algorithm is presented for the k-cut problem in graphs without vertex weights and fixed k. [16] presents an integer linear programming model for the problem, which can be solved by a branch-and-cut algorithm. Such a model is extended in [9] to graphs with vertex weights and capacity constraints for subdomains (a.k.a. node capacitated problem). As shown here, the resulting programming model is symmetrical in the subdomains and a simple heuristic based on fixing vertices in subdomains is proposed there to reduce model symmetry. In [23] and [10], separation heuristics are proposed to add valid inequalities to the model to improve the efficiency of enumeration algorithms. [24] presents a linear time algorithm that removes redundant parts of the branch-and-cut tree produced by the symmetry. In [25], column generation and a specialized branching technique are proposed for the capacitated graph partitioning problem. In [26], an improved branch-and-bound algorithm is presented for exact graph partitioning based on lower bounds for the multicommodity flow problem. Nevertheless, most exact methods have problems to solve instances with more than a few hundreds of vertices and edges. Alternatively, a heuristic solution can

be obtained quickly by general-purpose optimisation algorithms, namely simulated annealing [27], tabu search [28], genetic algorithms [29] and linear programming [30]. It is worth noting that, although several approximation algorithms have been reported for uncapacitated versions of the problem (e.g., min k-cut [31], balanced min k-cut [32]), to the authors' knowledge, there is no such algorithm for the capacitated min k-cut problem.

The previous algorithms have been used in cellular networks to assign cells to base station controllers [33], mobile switching centers [13,34–36] and location areas [37–40,14,41,36,42–44]. These problems have several important differences with the problem considered here. The assignment of controllers and switches during network planning differs from the planning of PCUs in the inclusion of cabling costs from the backhaul network in the objective function. Moreover, when optimizing such assignments during network operation, operators have to keep the number of changes to a minimum, since any cell re-allocation requires changing the backhaul network. In [13], an integer programming model is proposed for the optimal assignment of cells to switches in greenfield design. The previous model is modified in [33] to improve an existing assignment of cells to BSCs. In the latter, model symmetry is reduced by taking advantage of the limited number of changes. Likewise, location area planning differs from the problem here in that: a) the number of subdomains is not fixed, b) the cost function has an additional term increasing with the number of vertices per subdomain (paging cost), and c) there is no need to restrict the weight imbalance ratio between subdomains. In [43], location area planning is modeled as a set covering problem and solved by a greedy algorithm, which is especially suited for dynamic re-planning of location areas. In [14], an integer programming model for location area planning is tested on planar graphs based on a regular grid. In [44], a pseudo-approximation algorithm based on region growing is proposed for location area planning. Such an algorithm achieves an approximation factor for the overall signaling cost of $c\ln(n+1)$, where n is the number of cells and c is an arbitrary constant, but to a problem (different from the original one) where weight constraints have been increased (i.e., relaxed) by a factor of $c/(c+1)$.

In large wireless ad-hoc networks, clustering is used to build a hierarchical structure to simplify network routing [45]. For this purpose, nearby nodes are aggregated into groups represented by a designated node known as cluster head. Some major differences compared to the problem considered here are that, in these networks, a) a node might belong to several subdomains, b) the subgraph of each subdomain must be connected, c) the number of subdomains is not fixed, d) distributed algorithms are preferred, and, most importantly, e) the cost to be minimized not only considers the edge cut (communication exchange between clusters), but also the number of hops of internal paths within a subdomain (delay), the distance between nodes in the same cluster (power consumption) and the number of articulation vertices (reliability). In [46], a two-step approach is proposed to structure an OSPF-based ad-hoc network. First, a multi-level graph partitioning algorithm is used to define balanced subdomains with minimal inter-domain traffic. Then, a heuristic algorithm tailors the solution to OSPF specific needs. In [47], a distributed graph partitioning algorithm is presented to find optimal k-way partitions with respect to a broad range of cost functions in a wireless sensor network. In [48], an approximate graph partitioning algorithm is proposed to build perfectly balanced and connected subdomains in a wireless sensor network. All these algorithms are not applicable to the problem considered here.

3. Problem Formulation

In the CPAP, the aim is to minimize the number of cell re-selections (CRs) between cells in different PCUs, while keeping the load of all PCUs in the BSC within certain limits. This problem can be formulated as a graph partitioning problem [5]. In this approach, the BSC service area is modeled by an undirected weighted graph, whose vertices represent the cells in the BSC and edges represent the adjacencies between cells. The weight of each vertex denotes its contribution to the load of the PCU in terms of the number of time slots (TSLs) devoted to data traffic, while the weight of each edge denotes the number of users performing a CR between cells on its ends, which can be estimated directly from handover statistics [3]. The partitioning of the graph performed by grouping vertices into a fixed number of disjoint subsets, k, referred to as *subdomains*, reflects the assignment of cells to PCUs. Any partition defines a set of edges that join vertices in different subdomains, referred to as a *multi-cut*, δ, which represents those adjacencies between cells on different PCUs. Thus, the quality of a partition is defined by the sum of the weights of edges in the multi-cut, referred to as *edge cut*, and the ratio of weights of the most heavily loaded and the most lightly loaded subdomains, referred to as *weight imbalance ratio*.

The CPAP can be modeled as a *bounded, min-k cut problem* [21], described as follows. Let $G = (V, E)$ be an undirected weighted graph, consisting of a set of vertices V and edges E. Let ω_i be the weight of vertex i and γ_{ij} the weight of edge (i,j). Let B_{aw} and B_{rw} be real numbers defined as absolute and relative weight bounds, such that $0 < B_{aw} \leq \sum_{i \in V} \omega_i$ and $1 < B_{rw} < \infty$. The problem stands for the partition of V into k subdomains, V_1, V_2, ..., V_k, such that

$$\|V_n\| \triangleq \sum_{i \in V_n} \omega_i \leq B_{aw} \qquad \forall\, n \in \{1, 2, \ldots, k\} \tag{1}$$

(i.e., the weight of each subdomain is bounded),

$$\frac{\max(\|V_1\|, \ldots, \|V_k\|)}{\min(\|V_1\|, \ldots, \|V_k\|)} \leq B_{rw} \tag{2}$$

(i.e., the weight imbalance ratio is bounded) and

$$\|\delta(V_1,\ldots,V_k)\| \triangleq \sum_{(i,j)\in\delta(V_1,\ldots,V_k)} \gamma_{ij} \tag{3}$$

(i.e., the edge cut) is minimized. This formulation differs from the classical formulation of the graph partitioning problem in constraint (2), which is imposed by operators. Such a constraint avoids that some PCUs are overloaded, while others are underutilized. By fully exploiting the overall capacity of the existing set of PCUs, a more future-proof network structure is obtained, which can deal with network and traffic growth. Note that, if the overall PCU capacity in the network is not used, it is more likely that the addition of new base stations causes the need for re-allocating many other base stations to a different controller (or the need for adding new controllers). Also note that (2) does not control the maximal load of the controllers, which is done by (1).

The previous problem can be formulated as an ILP model, for which several different formulations exist. In contrast to Linear Programming (LP), the selection of a good model in ILP is of crucial importance for solving the problem efficiently. In the following paragraphs, three different ILP models are presented for the CPAP as a result of different actions to simplify or improve the initial formulation. Hereafter, $|V|$ is the number of cells in the BSC, $|E|$ is the number of adjacencies in the BSC and N is the set of PCUs in the BSC, $\{1, 2, \cdots, k\}$.

3.1. *General Model*

A natural starting point is to define variables describing which cells and adjacencies lie within each PCU [13]. Let X_{in} be binary variables that reflect the decision to assign cell i to PCU n, such that

$$X_{in} = \begin{cases} 1 & \text{if cell } i \text{ is assigned to PCU } n, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Let Y_{ij} also be binary variables that reflect whether adjacency (i,j) does not contribute to the edge cut, such that

$$Y_{ij} = \begin{cases} 1 & \text{if cell } i \text{ and } j \text{ belong to the same PCU,} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The single-homing constraint forces that each cell must be assigned to only one PCU, and therefore

$$\sum_{n\in N} X_{in} = 1 \qquad \forall\, i \in V. \tag{6}$$

The maximum number of data TSLs in a PCU is limited by physical hardware capabilities, leading to a constraint on the maximum subdomain weight as

$$\sum_{i\in V} \omega_i X_{in} \le B_{aw} \qquad \forall\, n \in N. \tag{7}$$

Following operator demand, the load must be evenly balanced among PCUs. Thus, a maximum weight imbalance is permitted between PCUs. This constraint can be expressed as

$$\frac{\sum_{i\in V} \omega_i X_{im}}{\sum_{i\in V} \omega_i X_{in}} \le B_{rw} \qquad \forall\, m,n \in N,\ m \ne n, \tag{8}$$

which can be transformed into the linear constraint

$$\sum_{i\in V} \omega_i X_{im} - B_{rw} \sum_{i\in V} \omega_i X_{in} \le 0 \qquad \forall\, m,n \in N,\ m \ne n. \tag{9}$$

The variables X_{in} and Y_{ij} are not independent, but are logically connected by a conjunctive operator in a constraint of the form

$$Y_{ij} = \sum_{n\in N} X_{in} X_{jn} \qquad \forall\, (i,j) \in E. \tag{10}$$

Thus, Y_{ij} equals 1 only if X_{in} and X_{jn} are both 1 for any n (i.e., cells i and j belong to the same PCU). Constraint (10) is not linear, but quadratic. The problem is thus a quadratic integer programming problem, which can not be solved by standard ILP techniques. However, it is possible to convert (10) into a linear program by extending the variable and constraint set [13]. Let Z_{ijn} be new binary variables defined as

$$Z_{ijn} = X_{in} X_{jn} \qquad \forall\, (i,j) \in E,\ n \in N, \tag{11}$$

which equals 1 if cells i and j are assigned to PCU n, 0 otherwise. This non-linear constraint can be disaggregated into the set of linear constraints

$$Z_{ijn} \le X_{in}, \qquad \forall\, (i,j) \in E,\ n \in N, \tag{12}$$
$$Z_{ijn} \le X_{jn}, \qquad \forall\, (i,j) \in E,\ n \in N, \tag{13}$$
$$Z_{ijn} \ge X_{in} + X_{jn} - 1, \qquad \forall\, (i,j) \in E,\ n \in N, \tag{14}$$
$$Z_{ijn} \ge 0, \qquad \forall\, (i,j) \in E,\ n \in N. \tag{15}$$

The objective function can be interchangeably defined as

$$\text{Min} \quad \sum_{i\in V} \sum_{j\in V} \gamma_{ij} (1 - \sum_{n\in N} Z_{ijn}) \tag{16}$$

(i.e., minimize the number of handovers between cells in different PCUs) or

$$\text{Max} \quad \sum_{i\in V} \sum_{j\in V} \gamma_{ij} \sum_{n\in N} Z_{ijn} \tag{17}$$

(i.e., maximize the number of handovers between cells in the same PCU).

The resulting model, denoted as *general model* (GM), is summarized as follows

$$\text{Max} \sum_{(i,j)\in E} \gamma_{ij} \sum_{n\in N} Z_{ijn} \tag{18}$$

$$\text{s.t.} \sum_{n\in N} X_{in} = 1, \qquad \forall\, i \in V, \tag{19}$$

$$\sum_{i\in V} \omega_i X_{in} \le B_{aw}, \qquad \forall\, n \in N, \tag{20}$$

$$\sum_{i\in V} \omega_i (X_{im} - B_{rw} X_{in}) \le 0, \quad \forall\, m,n \in N,\ m \ne n, \tag{21}$$

$$Z_{ijn} \le X_{in}, \qquad \forall\,(i,j) \in E,\ \forall\, n \in N, \tag{22}$$

$$Z_{ijn} \le X_{jn}, \qquad \forall\,(i,j) \in E,\ \forall\, n \in N, \tag{23}$$

$$Z_{ijn} \ge X_{in} + X_{jn} - 1, \qquad \forall\,(i,j) \in E,\ \forall\, n \in N, \tag{24}$$

$$X_{in} \in \{0,1\}, \qquad \forall\, i \in V,\ \forall\, n \in N, \tag{25}$$

$$Z_{ijn} \in \{0,1\}, \qquad \forall\,(i,j) \in E,\ \forall\, n \in N, \tag{26}$$

where X_{in} y Z_{ijn} are binary variables reflecting the assignment of base station i and adjacency (i,j) to PCU n, respectively. (18) reflects the goal of maximizing the number of users moving between base stations assigned to the same PCU. (19) ensures that any base station is assigned to a single PCU. (20) reflects the capacity limit of the PCU in terms of the maximum number B_{aw} of TSLs dedicated to packet-data services, while (21) enforces that the load is evenly balanced between PCUs by limiting the maximum imbalance ratio B_{rw} between PCUs. (22)–(24) show the dependency between decision variables by means of linear constraints and (25)–(26) are the binary constraints.

It can easily be deduced that the number of variables and constraints in GM is

$$N_{var\,GM} = k\,(|V| + |E|)\,, \tag{27}$$

$$N_{const\,GM} = N_{var\,GM} + |V| + k + 2\sum_{n=1}^{k-1} n + 3k|E|\,, \tag{28}$$

where k is the number of PCUs, $|V|$ is the number of base stations in the BSC and $|E|$ is the number of bi-directional adjacencies. In (28), note that $N_{var\,GM}$ out of $N_{const\,GM}$ constraints are integrality constraints.

3.2. *General Models without Symmetry*

The model GM necessarily has multiple optimal solutions, since any interchange in the PCU index n leads to indistinguishable solutions. This symmetry in the model is known to degrade the performance of enumeration algorithms used to solve the ILP model [49]. To avoid redundant solutions, a new set of $k-1$ constraints can be included to force the number of base stations in the PCUs not to increase with the PCU index [49,12]. The resulting model, referred to as *general model without symmetry 1* (GMS1), is

$$\text{Max} \sum_{(i,j)\in E} \gamma_{ij} \sum_{n\in N} Z_{ijn} \tag{29}$$

$$\text{s.t.} \sum_{n\in N} X_{in} = 1, \qquad \forall\, i \in V, \tag{30}$$

$$\sum_{i\in V} \omega_i X_{in} \le B_{aw}, \qquad \forall\, n \in N, \tag{31}$$

$$\sum_{i\in V} \omega_i (X_{im} - B_{rw} X_{in}) \le 0, \ \forall\, m,n \in N,\ m \ne n, \tag{32}$$

$$\sum_{i\in V} (X_{in} - X_{i,n+1}) \ge 0, \qquad \forall\, n \in N,\ n \ne k, \tag{33}$$

$$Z_{ijn} \le X_{in}, \qquad \forall\,(i,j) \in E,\ \forall\, n \in N, \tag{34}$$

$$Z_{ijn} \le X_{jn}, \qquad \forall\,(i,j) \in E,\ \forall\, n \in N, \tag{35}$$

$$Z_{ijn} \ge X_{in} + X_{jn} - 1, \qquad \forall\,(i,j) \in E,\ \forall\, n \in N, \tag{36}$$

$$X_{in} \in \{0,1\}, \qquad \forall\, i \in V,\ \forall\, n \in N, \tag{37}$$

$$Z_{ijn} \in \{0,1\}, \qquad \forall\,(i,j) \in E,\ \forall\, n \in N, \tag{38}$$

where the model symmetry is broken by simply adding the new constraints (33).

Alternatively, the symmetry can be reduced by assigning any one base station (e.g., v) to any one PCU (e.g., 1) [9]. This is easily accomplished by adding the constraint $X_{v1} = 1$. Fixing the assignment of a particular cell to a PCU implies that the set of variables $\{X_{vn} : n \in N\}$ and $\{Z_{vjn} : (v,j) \in E,\ n \in N\}$ can be deleted from the model, together with the constraints on these variables. To reduce symmetry further, a new set of $k-2$ constraints is included to force the number of cells in the remaining PCUs not to increase with the PCU index [49,12]. Formally, the new model, denoted as *general model without symmetry 2* (GMS2), is described as

$$\text{Max} \sum_{j\in V(v)} \gamma_{vj} X_{j1} + \sum_{(i,j)\in E-E(v)} \gamma_{ij} \sum_{n=1}^{k} Z_{ijn} \tag{39}$$

$$\text{s.t.} \sum_{n\in N} X_{in} = 1, \qquad \forall\, i \in V,\ i \ne v, \tag{40}$$

$$\sum_{i\in V,\, i\ne v} \omega_i X_{i1} \le B_{aw} - \omega_v, \tag{41}$$

$$\sum_{i\in V} \omega_i X_{in} \le B_{aw}, \qquad \forall\, n \in N,\ n \ne 1, \tag{42}$$

$$\sum_{i\in V,\, i\ne v} \omega_i X_{i1} + \omega_v - B_{rw} \sum_{i\in V,\, i\ne v} \omega_i X_{in} \le 0, \qquad \forall\, n \in N,\ n \ne 1, \tag{43}$$

$$\sum_{i\in V,\, i\ne v} \omega_i X_{in} - B_{rw} \left(\sum_{i\in V,\, i\ne v} \omega_i X_{i1} + \omega_v \right) \le 0, \qquad \forall\, n \in N,\ n \ne 1, \tag{44}$$

$$\sum_{i\in V,\, i\ne v} \omega_i X_{im} - B_{rw} \sum_{i\in V,\, i\ne v} \omega_i X_{in} \le 0, \qquad \forall\, m,n \in N,\ m,n \ne 1,\ m \ne n, \tag{45}$$

$$\sum_{i\in V} (X_{in} - X_{i,n+1}) \ge 0, \qquad \forall\, n = 2,...,k-1, \tag{46}$$

$$Z_{ijn} \leq X_{in}, \qquad \forall\,(i,j) \in E-E(v), \forall\, n \in N, \tag{47}$$
$$Z_{ijn} \leq X_{jn}, \qquad \forall\,(i,j) \in E-E(v), \forall\, n \in N, \tag{48}$$
$$Z_{ijn} \geq X_{in} + X_{jn} - 1, \forall\,(i,j) \in E-E(v), \forall\, n \in N, \tag{49}$$
$$X_{in} \in \{0,1\}, \qquad \forall\, i \in V,\ i \neq v,\ \forall\, n \in N, \tag{50}$$
$$Z_{ijn} \in \{0,1\}, \qquad \forall\,(i,j) \in E-E(v), \forall\, n \in N, \tag{51}$$

where $V(v)$ and $E(v)$ are the neighbor cells and adjacencies of cell v, respectively. Note that, compared to GMS1, (31) is split into (41) and (42), (32) is split into (43)-(45) and (33) is substituted by (46).

The number of variables and constraints in GMS1 and GMS2 is

$$N_{var\,GMS1} = k\,(|V| + |E|)\,, \tag{52}$$
$$N_{const\,GMS1} = N_{var\,GMS1} + |V| + k + 2\sum_{n=1}^{k-1} n + (k-1) + 3k|E|\,, \tag{53}$$
$$N_{var\,GMS2} = k(|V| - 1 + |E| - |E(v)|), \tag{54}$$
$$N_{const\,GMS2} = N_{var\,GMS2} + |V| + k + 2\sum_{n=1}^{k-1} n + (k-2) + 3k(|E| - |E(v)|)\,. \tag{55}$$

Note that GMS2 has slightly less variables and constraints than GMS1.

3.3. *Alternative Model*

The number of variables in the model can be reduced by evaluating only whether both source and target base station in an adjacency are assigned to the same PCU (i.e., whether an edge is not in the edge cut), regardless of to which PCU are assigned. Thus, variables Z_{ijn} are substituted by variables Y_{ij}, such that $Y_{ij} = 1$ if base stations i and j are in the same PCU, 0 otherwise. The resulting model, obtained by modifying (39) and (47)–(49) in GMS2, is

$$\text{Max} \sum_{j \in V(v)} \gamma_{vj} X_{j1} + \sum_{(i,j) \in\, E-E(v)} \gamma_{ij} Y_{ij} \tag{56}$$
$$\text{s.t.} \sum_{n \in N} X_{in} = 1, \qquad \forall\, i \in V,\ i \neq v, \tag{57}$$
$$\sum_{i \in V,\, i \neq v} \omega_i X_{i1} \leq B_{aw} - \omega_v, \tag{58}$$
$$\sum_{i \in V} \omega_i X_{in} \leq B_{aw}, \qquad \forall\, n \in N,\ n \neq 1, \tag{59}$$
$$\sum_{i \in V,\, i \neq v} \omega_i X_{i1} + \omega_v - B_{rw} \sum_{i \in V,\, i \neq v} \omega_i X_{in} \leq 0, \qquad \forall\, n \in N,\ n \neq 1, \tag{60}$$
$$\sum_{i \in V,\, i \neq v} \omega_i X_{in} - B_{rw} \left(\sum_{i \in V,\, i \neq v} \omega_i X_{i1} + \omega_v \right) \leq 0, \qquad \forall\, n \in N,\ n \neq 1, \tag{61}$$
$$\sum_{i \in V,\, i \neq v} \omega_i X_{im} - B_{rw} \sum_{i \in V,\, i \neq v} \omega_i X_{in} \leq 0, \qquad \forall\, m,n \in N,\ m,n \neq 1,\ m \neq n, \tag{62}$$
$$\sum_{i \in V} (X_{in} - X_{i,n+1}) \geq 0, \qquad \forall\, n = 2, ..., k-1, \tag{63}$$
$$Y_{ij} \geq X_{in} + X_{jn} - 1, \ \forall\,(i,j) \in E-E(v), \forall\, n \in N, \tag{64}$$
$$Y_{ij} \leq X_{in} - X_{jn} + 1, \ \forall\,(i,j) \in E-E(v), \forall\, n \in N, \tag{65}$$
$$Y_{ij} \leq X_{jn} - X_{in} + 1, \ \forall\,(i,j) \in E-E(v), \forall\, n \in N, \tag{66}$$
$$X_{in} \in \{0,1\}, \qquad \forall\, i \in V,\ i \neq v,\ \forall\, n \in N, \tag{67}$$
$$Y_{ij} \in \{0,1\}, \qquad \forall\,(i,j) \in E-E(v). \tag{68}$$

The number of variables and constraints in this model, referred to as *alternative model* (AM), is

$$N_{var\,AM} = k\,(|V| - 1) + |E| - |E(v)|\,, \tag{69}$$
$$N_{const\,AM} = N_{var\,AM} + |V| + k + 2\sum_{n=1}^{k-1} n + (k-2) + 3k(|E| - |E(v)|)\,. \tag{70}$$

From (54) and (69), it can be deduced that AM has $(k-1)(|E| - |E(v)|)$ less variables than GMS2.

4. Solution Algorithm

The algorithm used by most optimisation packages to solve integer programming problems is the *branch-and-bound* (BB) algorithm [50]. This algorithm follows an enumerative approach based on the 'divide and conquer' principle. The method starts by considering the original problem, for which lower and upper bounds for the optimal value are computed. This is done by solving the LP-relaxation of the problem, where the integrality constraints are removed. If both bounds coincide, an optimal solution has been found for the problem and the procedure terminates. Otherwise, the original problem is subdivided into several subproblems. This is accomplished by subdividing the space of feasible solutions into two or more regions, which together cover the whole feasible space. The algorithm is

applied recursively to the subproblems, generating a tree of subproblems. If an optimal solution is found to any of these subproblems, it is a feasible solution to the full problem, but it is not necessarily globally optimal. These feasible solutions can be used to prune the rest of the tree based on the bounds of the other problems. The search proceeds until all nodes in the tree have been solved or pruned, or until some specified threshold is met between the best value found so far and the lower bounds on all unsolved subproblems.

The BB algorithm is often combined with *cutting planes* methods [15], which is commonly referred to as *branch-and-cut* (BC) algorithm. The main idea behind cutting planes is to add constraints to the LP-relaxation of the problem until the optimal feasible solution takes on integer values. These additional constraints (referred to as *cuts*) must ensure that every feasible integer solution for the original problem is feasible for the cut and the current fractional solution is not feasible for the cut. Thus, a better lower bound for the problem can potentially be achieved. Cutting planes are iteratively added until either an integer solution is found or it becomes impossible or too expensive to find another cutting plane. In the latter case, a traditional branch operation is performed and the search for cutting planes continues on the subproblems. The use of cutting planes reduces the number of nodes in the tree, which improves the efficiency of the enumeration process.

To speed up the search, an initial heuristic solution (referred to as MIP start) can be fed to the BC algorithm. This solution gives a bound to fix variables and to discard branches within the search tree. Even though a solution may be currently configured in the network, it does not necessarily satisfy the formulated constraints. On the contrary, field trials have shown that solutions currently implemented in the network often result in a large load imbalance amongst PCUs of the same BSC [3]. Therefore, the existing solution is not used in this work. Instead, a *multi-level refinement* (ML) algorithm [7] is used to build an initial feasible solution from scratch. This technique compares favorably to other heuristics both in solution quality and runtime. While traditional graph partitioning algorithms work directly on the original graph, ML first coarsens the graph by collapsing vertices and edges to reduce the size of the graph. Initial solutions are efficiently computed on smaller versions of the graph and later uncoarsened to obtain the partition of the original graph. After each uncoarsening step, the *Fiduccia-Mattheyses refinement* (FM) algorithm [51] is applied on small portions of the graph close to the partition boundary. This local search algorithm aims to minimize edge cut by moving vertices to a different subdomain, escaping from local minima by exploring movements that temporarily increase edge cut. The main reason for choosing ML is the fact that most graph partitioning codes in the public domain implement this algorithm. Hence, although other heuristic algorithms may give better solutions [6], their adoption would have led to a loss of generality in the analysis. Figure 1 shows the template of the implemented multi-level refinement algorithm.

1) **Coarsening stage**
   Repeat match vertices by *Heaviest Edge Matching* [52]:
   - rank edges based on decreasing weight by *Quicksort* algorithm, and
   - select next edge and match endvertices if not already matched, until no edge is left unchecked,

   until the number of vertices in the coarsest graph equals the minimum number of subdomains imposed by the maximum subdomain weight.
2) **Initial partitioning**
   Build an initial partition by assigning each vertex in the coarsest graph to a different subdomain.
3) **Uncoarsening stage**
   Repeat progressively refine the initial partition on the coarsest graph:
   - uncoarsen the coarser graph based on matching scheme, and
   - refine the current partition by *Fiduccia-Matheysses refinement*, [51]

   until the finer graph is the original graph.

Fig. 1. Template of the multi-level refinement algorithm.

Although the above-described method was initially conceived for benchmarking purposes, it has also been considered for daily use. In principle, an instance of the CPAP must be solved only when a new cell or PCU is added to a BSC (provided that user mobility trends have not changed). Such an event occurs, at most, a few times a week, which should give plenty of time to compute the optimal solution. However, even with such loose time constraints, the operator might be forced to stop the BC algorithm before completion due to its large computational load. In such an approach, BC is used as a heuristic search algorithm, as the optimal solution is not guaranteed. However, unlike pure heuristic methods, a lower bound of the optimal value is still available, i.e., the highest lower objective bound of problems in the BC tree. Thus, an upper bound of the impairment in solution quality due to the early stop can be obtained from the difference between the best known feasible integer solution and the best lower objective bound (known as *IP gap*).

5. Performance Analysis

The proposed method is used to solve the CPAP in a set of problem instances built from data of a live GERAN. For clarity, the analysis set-up is described first. Then, a preliminary analysis highlights several features of graphs in the CPAP that motivate the use of exact methods. Finally, performance results are discussed.

5.1. *Analysis Set-up*

The optimized network area covers a geographical area of 150000 km^2, comprising 8952 base stations distributed over 59 BSCs. As the CPAP is solved on a per-BSC basis, the set

of problem instances considered in the analysis consists of 59 graphs. Robust performance estimations are expected, since the collection of graphs covers a large geographical area with very different propagation and mobility features.

The input to the method consists of data in the network management system and optimization constants. The handover statistics for a 2-week period (γ_{ij}), the number of data TSLs per cell (ω_i) and the number of PCUs per BSC (k) are used to build the graphs. As optimization constants, the maximum number of TSLs per PCU (B_{aw}) is set to 256 due to hardware limitations and the maximum load imbalance ratio between PCUs (B_{rw}) is set to 2 following the operator demand.

During the analysis, several graph partitioning methods are compared. The main concern is the exact method that applies BC with a MIP start from ML to an ILP model of the problem (denoted as BC). In BC, the four above-described formulations are compared: the classical general model (GM), the two models that break model symmetry in different ways (GMS1 and GMS2) and the alternative model (AM), which reduces the number of variables by only considering a single variable per adjacency. For comparison purposes, four heuristic approaches are also considered: a) the current manual approach followed by the operator (denoted as OI, for *operator initial*), b) the ML algorithm [7], used to initialize BC, since it is currently the benchmark used in the literature, c) the *Greedy Graph Growing Partitioning* (GGGP) algorithm [17], which builds subdomains around randomly selected seed vertices and performs several attempts (denoted as RGGGP, for randomized), and d) the *evolutionary multi-level search* (EMLS) algorithm [18], which applies ML to a population of graphs whose vertices and edges are biased depending on the quality of previous partitions. Results in [18] show that the latter outperforms state-of-the-art partitioning packages when constructing very high quality partitions for benchmarking purposes.

For efficiency, the exact method was implemented from routines in the public domain. The exact method is based on the BC algorithm in Gurobi Optimizer 4.0.0 [53] (preliminary results with other solvers can be found in [54]). In contrast, ML, RGGGP and EMLS were developed from scratch in MATLAB®v7.5.

The different algorithms were run on a dual-core computer under Windows with a clock frequency of 2.8GHz and 2GByte of RAM. In the exact approach, the 59 problem models were written in *Mathematical Programming System* (MPS) format and then fed to Gurobi. Gurobi was configured with default settings and no prior knowledge was assumed about the problem structure (apart from the heuristic solution). All problem instances were run to optimality (i.e., until the best solution is proved to be optimal).

Several performance indicators are evaluated during the assessment process. From the operator point of view, the main figure of merit is the total number of inter-PCU handovers obtained by the new PCU plan in the whole area, given by the sum of edge cuts in the 59 problem instances.

Table 1
Main statistics of the scenario.

Parameter	Avg	Std	Min	Max
Nbr. of cells per BSC	144.9	26.1	85	213
Nbr. of PCUs per BSC	5.3	0.9	4	8
Nbr. of adjacencies per BSC	899.8	282.9	241	1547
Nbr. of data TSLs per BSC	357.9	75.1	208	593
Avg. nbr. of cells per PCU	27.7	4.5	15.9	37.4
Avg. nbr. of data TSLs per PCU	68.0	11.5	41.6	91.0
Nbr. of adjacencies per cell	12.4	7.2	0	48
Nbr. of data TSLs per cell	2.5	1.1	1	8
Nbr. of handovers per adjacency	4301	14216	1	687208

The total running time is considered by operators as a secondary criteria, since PCU re-planning is only carried out a few times a year. However, from the academic side, it is interesting to evaluate the different formulations and tools, identifying those of a better computational efficiency.

5.2. *Preliminary Analysis of CPAP instances*

Table 1 presents relevant statistics of the analyzed network area. For clarity, statistics have been broken down in a BSC, PCU, cell and adjacency level (in graph notation, problem instance, subdomain, vertex and edge, respectively). A close inspection of the table reveals some significant differences with graph partitioning problems in other application domains. The analysis first focuses on graph attributes and then on optimization constraints.

Firstly, the size of graphs in the CPAP is not very large, since the number of vertices per graph (i.e., cells per BSC) is several orders of magnitude smaller than in other fields. While graphs of hundreds of thousands of vertices are often reported in supercomputing [8], VLSI design [55] and internetworking [21], CPAP graphs comprise only a few hundreds of vertices. Even if all the instances of the CPAP (i.e., BSCs in the network) are taken into account, the problem cannot be considered of extreme size, since problem complexity only grows linearly with the number of instances. Hence, unlike in other applications, exact methods are feasible for the CPAP. Regarding the number of subdomains per instance (i.e., PCUs per BSC), despite its small absolute value, this figure is high when compared to the number of vertices in the graph. As a consequence, the average number of vertices per subdomain (i.e., cells per PCU) is one order of magnitude below the values commonly reported in other fields. This property affects the performance of heuristic graph partitioning algorithms for several reasons. On the one hand, the benefit from ML approaches decreases, as the original graph cannot be simplified further during the coarsening stage. Thus, the fewer vertices in the graph, the fewer coarsening levels are required. On the other hand, refinement algorithms have less degrees of freedom, since only a small number of vertices can be moved without affecting the balance between subdomains. This problem is exacerbated by the heterogeneity of vertex weights (i.e., TSLs per cell), the largest of which can be up to eight times that of the smallest, as observed in Table 1.

At the same time, the edge density of CPAP graphs is small. An average of 12.4 adjacencies are defined per cell, which is much less than the average number of cells in the BSC (i.e. 144.9). As a result, the average density is 0.09. However, density figures are still larger than values reported in the graph partitioning literature. While the set of neighbors of a vertex in supercomputing graphs normally consists of 3–8 vertices, some vertices in CPAP graphs have up to 48 neighbors. Hence, although CPAP graphs can be broadly classified as sparse, edge density is larger than in other applications. This property is known to deteriorate the efficiency of most graph partitioning algorithms, whether exact [10,12] or heuristic [56].

The large number of neighbors per vertex makes that, in contrast to what has been assumed in previous work [13,14], graphs derived from user mobility in a cellular network are seldom planar. This property, together with graph heterogeneity, is expected to degrade the performance of local refinement algorithms. Likewise, the fourth column in Table 1 shows that some cells do not have adjacencies, becoming isolated vertices in the graph. This fact complicates the design of heuristic algorithms, which often assume that the original graph is connected.

The differences are not only restricted to graphs, but also extend to the optimization constraints. On the one hand, the definition of the imbalance constraint in other application domains differs from the way it is currently understood by cellular network operators. In most graph partitioning applications, the imbalance is defined as the ratio of the weight of the largest subdomain to the weight of a subdomain under perfect balance. In contrast, cellular operators are more interested in the ratio between the maximum and minimum subdomain weights (i.e., the weight imbalance ratio). Although it might seem that both definitions provide similar results, it is worth noting that the former only entails the control of the largest subdomain in the partition, whereas the latter also requires the control of the smallest subdomain. This complicates the design of heuristic methods and adds new constraints to the ILP model, which increases problem complexity. At the same time, the maximum weight imbalance ratio is much larger than in other applications. While imbalance ratios of 1.02–1.05 are normally targeted in supercomputing and VLSI design, a weight imbalance ratio of 2 is permitted in the CPAP. This is mainly due to the fact that, unlike in other applications, system performance is not severely affected by uneven load distribution. As a result of relaxing the imbalance constraint, the size of the feasible solution space increases, which degrades the efficiency of both heuristic and exact approaches. In this regard, it is worth mentioning that the binding constraint is the imbalance constraint, and not the absolute weight constraint. This can easily be deduced from Table 1, since the average number of TSLs per PCU (i.e., 68.2) is far less than the capacity limit of existing PCUs (i.e., 256).

All these features suggest that instances of the graph partitioning problem in the CPAP are of modest size, but harder to solve than instances of the same size in other application domains. These facts justify the use of more sophisticated methods, among which are exact methods.

5.3. *Performance Results*

The first experiment proves the capability of exact methods to find the best solution. Table 2 presents the results of the methods in the entire network area. For a fair comparison, the number of attempts in RGGGP and generations in EMLS (the same for all instances) is configured so as to obtain a total running time similar to that of the exact method (i.e., BC with GMS2). In the table, it is observed that the total number of inter-PCU handovers in the optimal solution (obtained by BC) is 81% smaller than that of the manual operator solution, denoted as OI. This is a clear indication of the need for optimizing PCU plans. It is also clear that ML is a very efficient heuristic for this type of graphs, since it achieves a three-fold reduction in the total number of inter-PCU handovers in a few seconds. Nonetheless, the number of inter-PCU handovers of ML is still 54% larger than that of BC. Obviously, such a low edge cut is obtained by BC at the expense of a much longer runtime. Gurobi needs half an hour per BSC on average when solving model GMS2. Not shown is the fact that such an average is dominated by the 5 largest instances, which take half of the total running time. Specifically, Gurobi manages to solve 52 out of the 59 instances in less than 1 hour. Note that RGGGP and EMLS seldom find the optimal solution in that time.

Having shown the potential of exact approaches, the next step is to identify the best ILP formulation of the CPAP. To save time, the analysis is restricted to the 10 smallest problem instances. Figure 2 shows the empirical cumulative density function (ecdf) of Gurobi runtime for the 10 smallest instances with the different models. Roughly speaking, the closer to the left, the faster the solver can process the model. In the figure, it is observed that GM is clearly the worst model. By eliminating model symmetry, GMS1 more than halves the running time. A four-fold reduction is obtained by combining different ways of breaking the symmetry as in GMS2. A deeper analysis reveals that GMS2 has a tighter LP relaxation, which is the origin of its superior performance. Specifically, the average gap between the ILP and LP solutions is 100% for GMS1 and 58% for both GMS2 and AM. Unexpectedly, AM performs worse than GMS2, even if the former has three times less variables and an LP relaxation as tight as the latter.

The efficiency of BC can be improved further by stopping it before completion. To show the benefits of this approach, Fig. 3 represents the time evolution of the edge cut of the best solution found by BC, RGGGP and EMLS in a real instance. In the example, RGGGP and EMLS obtain solutions better than that of ML quickly, but, as many other multi-start approaches, they stagnate after a few attempts. Even if EMLS stagnates later than RGGGP, the former also

Table 2
Performance of PCU plans built by different methods.

	OI	ML	RGGGP	EMLS	BC(GMS2)
Total no. of handovers [$\cdot 10^6$]	228.3	228.3	228.3	228.3	228.3
Total no. of inter-PCU handovers [$\cdot 10^6$]	75.7	22.3	14.4	13.8	13.1
Avg. PCU load imbalance ratio	3.59	1.67	1.94	1.96	1.99
No. of instances optimum found	0	1	3	5	59
No. of instances optimality proved	-	-	-	-	59
Total runtime [h]	-	0.005	35.2	30.0	28.3
Avg. runtime per instance [h]	-	10^{-4}	0.60	0.51	0.48

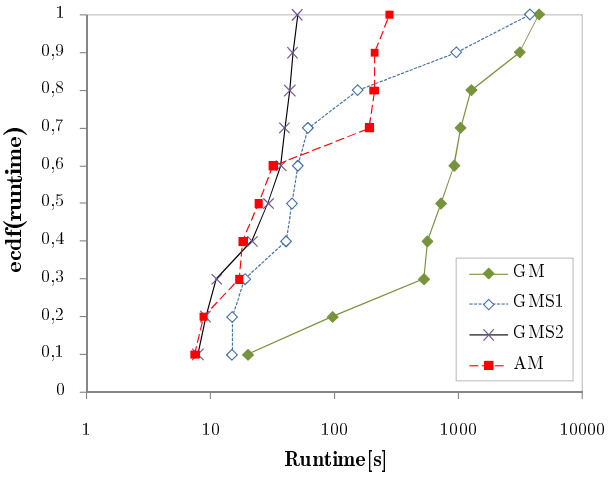

Fig. 2. Gurobi runtime over different formulations for a limited set of instances.

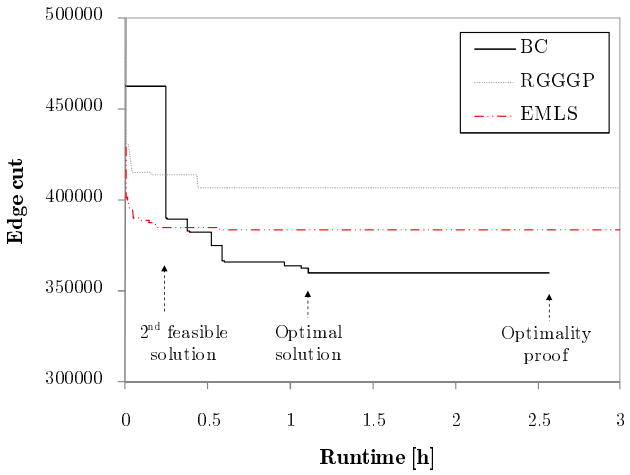

Fig. 3. Example of time evolution of best value found by the branch-and-cut algorithm in Gurobi in a real instance.

converges slowly to the optimal solution. In contrast, BC needs some time to find the second feasible solution (the first one with edge cut smaller than that of the ML solution), but, thereafter, BC progressively improves solution quality until the optimal solution is found. The remaining time is used to discard unexplored regions of the solution space to prove optimality. A comprehensive analysis of all instances shows that, overall, BC takes 10% of time to find the first feasible solution, 70% to find the optimal solution and 20% to prove optimality. From these values, it can be deduced that the execution time of BC can only be reduced slightly (i.e., 20%) without compromising solution quality. However, a larger reduction is still possible if some deterioration in solution quality is accepted. In this case, trading off solution quality for less runtime is beneficial because most of the improvement in solution quality is normally obtained at the beginning of the search process, as it is the case in the example of Fig. 3.

From Fig. 3, it is clear that, in order to use time efficiently, BC should run, at least, until the first feasible solution (different from the initial solution) is found. On the other hand, BC should run, at most, until the optimal solution is found. From the percentages reported above, it can be inferred that BC finds the first solution better than the ML solution in 3 minutes on average (i.e., 10% of the 0.48 hours of average runtime per instance), which is a reasonable time. However, other heuristic methods might make a more efficient use of time. Therefore, it is important to check if BC, when used as a heuristic method, is competitive with heuristic methods used for benchmarking purposes. For this purpose, the analysis evaluates the time required by BC to obtain a solution better than EMLS. Fig. 4 shows the ecdf of the time required by BC to outperform RGGGP and EMLS in terms of solution quality. It is observed that BC finds a solution better than EMLS in less than 15 minutes (900 seconds) in 52 out of the 59 instances (in all but one for RGGGP). From these results, it can be concluded that BC can be stopped before completion to obtain solutions of extreme quality under loose runtime constraints.

As other IP solvers, Gurobi displays the evolution of the relative IP gap during execution, giving the approximation ratio of the best solution found so far. Fig. 5 shows the evolution of the average and maximum relative IP gap in the 59 instances. It is observed that, after half an hour, the relative gap is below 6% in all instances. From the figure, it can be concluded that the approximation ratio of the algorithm is adequate and decreases steadily.

6. Conclusions

During network planning, cellular operators face clustering problems that can be solved by graph partitioning methods, among which is the assignment of cells to PCUs. In this work, it has been shown that this problem has several peculiarities that justify the use of exact methods. On

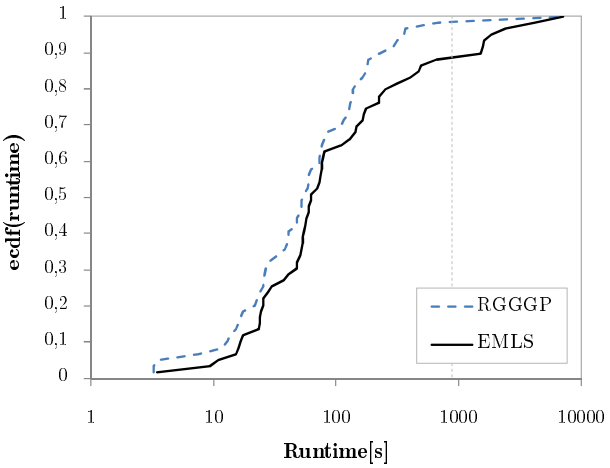

Fig. 4. Histogram of the time required by Gurobi to outperform the solution obtained by heuristic methods.

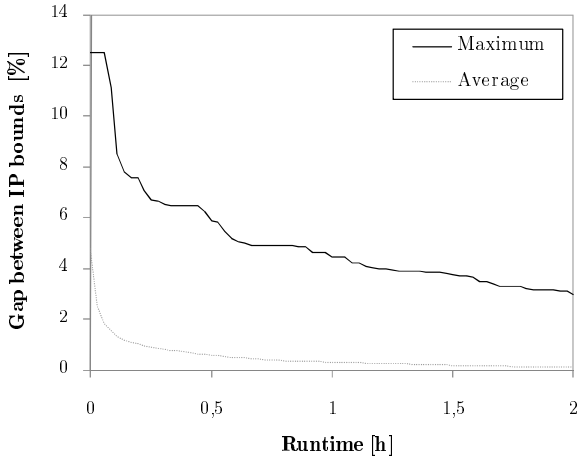

Fig. 5. Evolution of relative integer programming gap.

the one hand, the modest size of graphs makes exact approaches feasible. On the other hand, the small number of vertices and the heterogeneity of graphs degrade the effectiveness of classical heuristic approaches based on local refinement algorithms. These issues motivate the analysis of an exact method presented here.

An efficient integer linear programming model has been suggested for the problem. The main novelty is the reduction of symmetry in the model, which is known to degrade the performance of enumerative algorithms. Experiments have shown that the proposed model outperforms previous formulations of the graph partitioning problem in the cellular field.

The proposed model has been solved by the branch-and-cut algorithm in Gurobi, with a MIP start from a multi-level refinement algorithm. Performance assessment has been based on an extensive set of graphs built from data of a live GERAN. Results have shown that the proposed method clearly outperforms heuristic approaches in terms of solution quality. Based on handover statistics, it is estimated that the total inter-PCU cell re-selection ratio obtained by the exact method is 81% smaller than that of the operator solution and 35% smaller than that obtained by the multi-level refinement algorithm.

The main drawback of the exact method is its large execution time. In a number of cases, Gurobi had to be run for several hours to find the optimal solution and prove optimality. It should be pointed out that the size of problem instances considered here is larger than those reported in the literature (e.g., [13,14]), especially in the number of edges. To reduce the execution time, the branch-and-cut algorithm in Gurobi can be stopped before completion. The complexity of Gurobi makes it difficult (or impossible) to analyze the approximation ratio obtained by this approach under provable runtime bounds. In the absence of such a theoretical analysis, a performance analysis has shown that Gurobi is competitive under loose runtime constraints with heuristic methods used for benchmarking purposes. Gurobi solves 9 out of 10 instances in less than 1 hour, while finding a solution better than state-of-the-art multi-level approaches in the unsolved ones. Such an execution time is acceptable when re-planning a cellular network, which is performed locally in a limited geographical area and, at most, once a week.

With the steady increase of computing power, exact methods might become a worthy candidate for solving clustering problems in live cellular networks. From the academic side, it remains open whether general-purpose metaheuristics, such as simulated annealing and genetic algorithms, when used for graph partitioning might perform competitively over this kind of graphs. For a fair comparison, the latter algorithms should be compared with branch-and-cut algorithms specifically designed for the graph partitioning problem [10,26], which take full advantage of the structure of the problem. Note that, unlike the algorithms proposed here, no routines for these algorithms are available in the public domain, thus requiring a considerable coding effort. Finally, it is worth mentioning that, although the proposed method has been conceived for assigning cells to PCUs in GERAN, it can easily be adapted to similar clustering problems arising in other levels of network hierarchy or radio access technologies. For instance, in UMTS, user mobility statistics used to build network graphs are slightly different, since a terminal can be simultaneously connected to several base stations due to the soft handover feature. Likewise, in LTE-SAE, the ILP model has to consider that a base station can be simultaneously assigned to more than one access gateway.

Acknowledgements

This work has been supported by grant TIC2009-13413 from the Spanish Ministry of Science and Innovation.

References

[1] T. Halonen, J. Melero, J. Romero, GSM, GPRS and EDGE Performance: Evolution Toward 3G/UMTS, John Wiley & Sons,

2002.
[2] V. Rexhepi, M. Moissio, S. Hamiti, R. Vaittinen, Performance of streaming services in GERAN A/Gb mode, in: Proc. IEEE 60th Vehicular Technology Conference, Vol. 6, 2004, pp. 4511–4515.
[3] M. Toril, V. Wille, R. Barco, Optimization of the assignment of cells to packet control units in GERAN, IEEE Communications Letters 10 (3) (2006) 219 – 221.
[4] M. Garey, D. Johnson, Computers and Intractability: A Guide to NP-Completeness, W.H. Freeman and Company, California, 1979.
[5] K. Schloegel, G. Karypis, V. Kumar, Graph partitioning for high performance scientific simulations, in: J. Dongarra, I. Foster, G. Fox, W. Gropp, K. Kennedy, L. Torczon, A. White (Eds.), Sourcebook of parallel computing, Morgan Kaufmann, 2003.
[6] M. Toril, I. Molina, V. Wille, C. Walshaw, Analysis of heuristic methods for the assignment of packet control units in GERAN, Wireless Personal Communications, 2010 (available as Online First).
[7] G. Karypis, V. Kumar, Multilevel k-way partitioning scheme for irregular graphs, Journal of Parallel and Distributed Computing 48 (1) (1998) 96–129.
[8] C. Walshaw, M. Cross, Mesh partitioning: a multilevel balancing and refinement algorithm, SIAM Journal of Scientific Computing 22 (1) (2000) 63–80.
[9] S. Holm, M. Sorensen, The optimal graph partitioning problem, OR Spectrum 15 (1) (1993) 1–8.
[10] C. E. Ferreira, A. Martin, C. C. de Souza, R. Weismantel, L. A. Wolsey, The node capacitated graph partitioning problem: A computational study, Mathematical Programming 81 (2) (1998) 229–256.
[11] E. Macambira, C. Meneses, P. Pardalos, M. Resende, A novel integer programming formulation for the K-SONET ring assignment problem, Tech. Rep. TD-6HLLNR, AT&T Labs Research (Oct 2005).
[12] E. Macambira, N. Maculan, C. de Souza, A column generation approach for SONET ring assignment, Networks 47 (3) (2006) 157–171.
[13] A. Merchant, B. Sengupta, Assignment of cells to switches in PCS networks, IEEE/ACM Transactions on Networking 3 (5) (1995) 521–526.
[14] S. Ali, Location management in cellular mobile radio networks, in: Proceedings of 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, Vol. 2, 2002, pp. 745–749.
[15] G. Nemhauser, L. Wolsey, Integer and Combinatorial Optimization, John Wiley & Sons, 1999.
[16] S. Chopra, M. Rao, The partition problem, Mathematical Programming 59 (1) (1993) 87–115.
[17] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM Journal on Scientific Computing 20 (1) (1998) 359–392.
[18] A. Soper, C. Walshaw, M. Cross, A combined evolutionary search and multilevel optimisation approach to graph-partitioning, Journal of Global Optimization 29 (2) (2004) 225–241.
[19] Z. Wu, R. Leahy, An optimal graph theoretic approach to data clustering: theory and its application to image segementation, IEEE Transactions on pattern analysiss and machine intelligence 15 (1993) 1101–1113.
[20] G. Karypis, R. Aggarwal, V. Kumar, S. Shekhar, Multilevel hypergraph partitioning: applications in vlsi domain, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 7 (1) (1999) 69–79.
[21] R. Krishnan, R. Ramanathan, M. Steentrup, Optimization algorithms for large self-structuring networks, in: Proc. INFOCOM '99, Vol. 1, 1999, pp. 71–78.
[22] O. Goldschmidt, D. S. Hochbaum, A polynomial algorithm for the k-cut problem for fixed k, Mathematics of Operations Research 19 (1) (1994) 24–37.
[23] C. Ferreira, A. Martin, C. de Souza, R. Weismantel, L. Wolsey, Formulations and valid inequalities for the node capacitated graph partitioning problem, Math. Program. 74 (3) (1996) 247–266.
[24] V. Kaibel, M. Peinhardt, M. Pfetsch, Orbitopal Fixing, in: Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science, 2007, Ch. 7, pp. 74–88.
[25] A. Mehrotra, M. Trick, Cliques and clustering: A combinatorial approach, Operations Research Letters 22 (1) (1998) 1 – 12.
[26] N. Sensen, Lower bounds and exact algorithms for the graph partitioning problem using multicommodity flows, in: Europ. Symp. on Algorithms, 2001, Lecture Notes in Computer Science, Vol. 2161, 2001, pp. 391–403.
[27] D. Johnson, C. Aragon, L. McGeoch, C. Schevon, Optimization by simulated annealing: an experimental evaluation. Part I, graph partitioning, Operations Research 37 (6) (1989) 865–892.
[28] L. Tao, Y. Zhao, K. Thulasiraman, M. Swamy, Simulated annealing and tabu search algorithms for multiway graph partition, Journal of Circuits, Systems and Computers 2 (2) (1992) 159–185.
[29] T. Bui, B. Moon, Genetic algorithm and graph partitioning, IEEE Transactions on Computers 45 (7) (1996) 841–855.
[30] A. Lisser, F. Rendl, Graph partitioning using linear and semidefinite programming, Mathematical Programming 95 (2003) 91–101.
[31] V. V. Vazirani, Approximation Algorithms, Springer, 2003.
[32] G. Even, S. Rao, B. Schieber, Fast approximate graph partitioning algorithms, in: Proc. 8th ACM-SIAM Symposium on Discrete Algorithms, 1997, pp. 639–648.
[33] M. Toril, V. Wille, Optimization of the assignment of base stations to base station controllers in GERAN, IEEE Communications Letters 12 (6) (2008) 477–479.
[34] D. Saha, A. Mukherjee, P. Bhattacharjee, A simple heuristic for assignment of cells to switches in a PCS network, Wireless Personal Communications 12 (2000) 209–224.
[35] S. Pierre, F. Houeto, A tabu-search approach for assigning cells to switches in cellular mobile networks, Computer Communications 25 (5) (2002) 465–478.
[36] I. Demirkol, C. Ersoy, M. Caglayan, H. Delic, Location area planning and cell-to-switch assignment in cellular networks, IEEE Transactions on Wireless Communications 3 (3) (2004) 880–890.
[37] P. Gondim, Genetic algorithms and location area partitioning problem in cellular networks, in: Proc. 46th IEEE Vehicular Technology Conference, 1996, pp. 1835–1838.
[38] J. Plehn, The design of location areas in a GSM-network, in: Proc. 45th IEEE Vehicular Technology Conference, 1995, pp. 871–875.
[39] T. Wang, S. Hwang, C. Tseng, Registration area planning for PCS networks using genetic algorithms, IEEE Transactions on Vehicular Technology 47 (3) (1998) 987–995.
[40] P. Demestichas, E. Tzifa, V. Demesticha, N. Georgantas, G. Kotsakis, M. Kilanioti, M. Striki, M. Anagnostou, M. Theologou, Control of the location update and paging signaling load in cellular systems by means of planning tools, in: Proc. IEEE Vehicular Technology Conference, 1999, pp. 2119–2123.
[41] E. Cayirci, I. F. Akyildiz, Optimal location area design to minimize registration signaling traffic in wireless systems, IEEE Transactions on Mobile Computing 2 (1) (2003) 76–85.
[42] P. S. Bhattacharjee, D. Saha, A. Mukherjee, An approach for location area planning in a personal communication services network (PCSN), IEEE Transactions on Wireless Communications 3 (4) (2004) 1176–1187.
[43] S.-W. Lo, T.-W. Kuo, K.-Y. Lam, G.-H. Li, Efficient location area planning for cellular networks with hierarchical location databases, Computer Networks 45 (6) (2004) 715–730.

[44] Y. Bejerano, M. A. Smith, J. S. Naor, N. Immorlica, Efficient location area planning for personal communication systems, IEEE/ACM Transactions on Networking 14 (2) (2006) 438–450.
[45] A. Boukerche, Algorithms and Protocols for Wireless Sensor Networks, IEEE, 2008, Ch. Clustering in Wireless Sensor Networks: A Graph Theory Perspective, pp. 161 –193.
[46] S. Galli, H. Luss, J. Sucec, A. McAuley, S. Samtani, D. Dubois, K. DeTerra, R. Stewart, B. Kelley, A novel approach to OSPF-area design for large wireless ad-hoc networks, in: Proc. IEEE International Conference on Communications, Vol. 5, 2005, pp. 2986–2992.
[47] S. Roy, Y. Wan, A. Saberi, A flexible algorithm for sensor network partitioning and self-partitioning problems, in: S. Nikoletseas, J. Rolim (Eds.), Algorithmic Aspects of Wireless Sensor Networks, Vol. 4240 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2006, pp. 152–163.
[48] I. Slama, B. Jouaber, D. Zeghlache, Energy efficient scheme for large scale wireless sensor networks with multiple sinks, in: IEEE Wireless Communications & Networking Conference, 2008, pp. 2367–2372.
[49] H. Sherali, J. Smith, Improving discrete model representations via symmetry considerations, Management Science 47 (10) (2001) 1396–1407.
[50] A. Doig, A. Land, An automatic method of solving discrete programming problems, Econometrica 28 (1960) 497–520.
[51] C. Fiduccia, R. Mattheyses, A linear time heuristic for improving network partitions, in: Proc. 19th ACM/IEEE Design Automation Conference, 1982, pp. 175–181.
[52] A. Gupta, Fast and effective algorithms for graph partitioning and sparse matrix ordering, IBM Journal of Research and Development 41 (1/2) (1997) 171–184.
[53] Gurobi Optimization, Gurobi Optimizer Reference Manual Version 4.0 (Nov 2010).
[54] M. Toril, P. Guerrero-García, S. Luna-Ramírez, V. Wille, Re-assignment of base stations to packet controllers by integer programming, Tech. rep., COST2100 (Feb 2010).
[55] M. Junger, A. Martin, G. Reinelt, R. Weismantel, Quadratic 0/1 optimization and a decomposition approach for the placement of electronic circuits, Mathematical Programming 63 (3) (1994) 257–279.
[56] C. Walshaw, Multilevel refinement for combinatorial optimisation problems, Annals of Operations Research 131 (2004) 325–372.