Journal of Heuristics manuscript No.

An Adaptive Multi-start Graph Partitioning Algorithm for Structuring Cellular Networks

Matías Toril · Volker Wille · Iñigo Molina-Fernández · Chris Walshaw

Abstract In mobile network design, the problem of assigning network elements to controllers when defining network structure can be modeled as a graph partitioning problem. In this paper, a comprehensive analysis of a sophisticated graph partitioning algorithm for grouping base stations into packet control units in a mobile network is presented. The proposed algorithm combines multi-level and adaptive multi-start schemes to obtain high quality solutions efficiently. Performance assessment is carried out on a set of problem instances built from measurements in a live network. Overall results confirm that the proposed algorithm finds solutions better than those obtained by the classical multi-level approaches and much faster than classical multi-start approaches. The analysis of the optimization surface shows that the best local minima values follow a Gumbel distribution, which justifies the stagnation of naive multi-start approaches after a few attempts. Likewise, the analysis shows that the best local minima share strong similarities, which is the reason for the superiority of adaptive multi-start approaches. Finally, a sensitivity analysis shows the best internal parameter settings in the algorithm.

Keywords mobile network · optimization · graph partitioning · multi-level refinement · adaptive multi-start

This work has been supported by the Spanish Ministry of Science and Innovation (grant TEC2009-13413).

M. Toril · I. Molina-Fernández
Communications Engineering Dept., University of Málaga, Campus Universitario de Teatinos, s/n, E-29010, Málaga, Spain
Tel.: +34 952137120, Fax: +34 952132027, E-mail: {mtoril,imf}@ic.uma.es

V. Wille
Nokia Siemens Networks, Performance Services, Huntingdon, UK, E-mail: volker.wille@nsn.com

C. Walshaw
School of Computing and Mathematical Sciences, University of Greenwich, London, UK, E-mail: C.Walshaw@gre.ac.uk

1 Introduction

In the last decade, mobile telecommunications networks have become extremely complex and dynamic. As a result, efficient network management is key to providing high-quality services at a low operational cost. Until a few years ago, most operational tasks were carried out manually. However, network operators are currently showing a growing level of interest in automating most network management activities. This has stimulated intense research activities in the field of automatic mobile network management, (Laiho et al, 2002; Mishra, 2007).

One of the most time consuming tasks in mobile network management is the mere structuring of the network. Current mobile networks are given a hierarchical structure for scalability. During network design, network planners decide the cluster of elements in lower layers to be assigned to elements of a higher level (referred to as *controllers*). This clustering problem is repeated in the different layers of network hierarchy. In the past, this problem has been solved manually, leading to sub-optimal network performance in spite of the time and effort invested. Moreover, during network operation, the inclusion of new elements often forces a re-configuration of the network hierarchy to equalize the load of elements in higher layers. Hence, operators require automatic tools to solve the network structuring problem periodically and network wide.

The aim of clustering algorithms is two-fold: a) to avoid unnecessary signalling exchange by grouping closely related elements under the same controller, and b) to keep the load of network elements evenly balanced, since the overall network capacity is thus fully exploited. The underlying assignment problem can be formulated as a graph partitioning problem, (Schloegel et al, 2000). In this approach, the network is modeled as a graph, whose vertices and edges are the network elements and adjacency relationships between them, respectively. The partitioning of this graph into subdomains reflects the assignment of network elements to controllers. When partitioning the graph, the aim is to minimize the total weight of edges that join vertices in different subdomains, which represents the number of users moving between elements in different controllers.

The simplest approach to solve the graph partitioning problem is the enumeration of all possible solutions, which is impractical for graphs of reasonable size. Alternatively, the size of the solution space that must be explicitly enumerated can be reduced by describing the problem as an integer linear programming model, which is then solved by the Branch-and-Cut algorithm, (Ferreira et al, 1998). Nonetheless, these exact methods are still computationally intensive. Several methods have been proposed to find bounds, (Donath and Hoffman, 1973)(Lisser and Rendl, 2003), or approximate solutions efficiently (for a survey, see (Schloegel et al, 2000)). Amongst all, the Multi-Level Refinement algorithm, (Karypis and Kumar, 1998b), is the common benchmark against which more refined methods are compared. Other promising techniques based on meta-heuristics are Tabu Search, (Rolland et al, 1996), and Adaptive Multi-Start algorithms, (Hagen and Kahng, 1997). In the context of cellular networks, these heuristic methods have been applied to assign base stations to switches and location areas in network design, (Merchant and Sengupta, 1995; Gondim, 1996; Plehn, 1995; Saha et al, 2000; Pierre and Houeto, 2002; Demirkol et al, 2004; Bhattacharjee et al, 2004). However, limited attention has been paid to the properties of graphs in mobile network design and how these affect the performance of heuristic algorithms.

This paper deals with one such clustering problem, namely the assignment of base stations to Packet Control Units (PCUs) in a GSM-EDGE Radio Access Network (GERAN). In a previous paper, (Toril et al, 2006), this problem was formulated as a

graph partitioning problem and solved by a classical multi-level refinement algorithm. In this paper, a novel heuristic method for assigning PCUs is proposed. The new method combines two classical graph partitioning techniques that have been considered separately in the past: the multi-level refinement, (Hendrickson and Leland, 1995; Karypis and Kumar, 1998b; Walshaw and Cross, 2000), and the clustered adaptive multi-start, (Hagen and Kahng, 1997). Performance assessment is carried out by comparing the proposed method with the classical multi-level and multi-start approaches over an extensive set of graphs constructed from a live mobile network. To the authors' knowledge, no previous work has compared graph partitioning algorithms over real graphs from mobile network design. It is well known that the performance of graph algorithms strongly depends on graph attributes. It will be shown here that cellular graphs have important differences with those in supercomputing applications, for which most existing partitioning algorithms were designed, justifying the need for a separate analysis. The main contributions of this work are: a) an overall performance comparison of the algorithms under consideration for graphs from a live mobile network, b) a thorough analysis of the optimization surface of the problem to justify the limitations of naive multi-start approaches and the benefit of adaptive multi-start approaches, and c) a sensitivity analysis to determine the optimal value of internal parameter settings in the proposed adaptive multi-start algorithm. The rest of the paper is organized as follows. Section 2 formulates the problem of assigning base stations to packet control units in GERAN. Section 3 describes the proposed graph partitioning algorithm. Section 4 presents the results of the analysis over a set of graphs built with data from a live GERAN. Finally, Section 5 presents the main conclusions of the study.

2 Problem Formulation

Fig. 1 outlines the problem of assigning PCUs in a GERAN system. In a mobile network, the coverage area is divided into cells, each served by a single Base Station (BS), (Mouly and Pautet, 1992). BSs are assigned to a Base Station Controller (BSC), responsible for the control of a few hundred BSs. In GERAN, each BSC has a fixed number of PCUs, to which BSs must be assigned to offer packet data services. The assignment of BSs to PCUs during network design should minimize the number of users that change PCU after a change of BS, as service interruption is much longer in this case, which can degrade, sometime severely, the performance of packet data services, (Toril et al, 2006). Hence, BSs exchanging a large number of users between them should be assigned to the same PCU. At the same time, the teletraffic load must be evenly distributed among PCUs.

To perform the assignment of BSs to PCUs, the network area optimized (i.e., area serviced by a BSC) is modeled by a weighted undirected graph, $G = (V, E)$, as in Fig. 2. The vertices of the graph, V, represent the BSs in the BSC, while the edges, E, represent the adjacencies between BSs. The weight of a vertex, ω_i, represents the contribution of the BS to the PCU load in terms of the number of channels devoted to packet data services, while the weight of an edge, γ_{ij}, represents the number of users moving between BSs. As shown in (Toril et al, 2006), these quantities can be estimated from network statistics. The partition of the graph, performed by assigning the vertices to k subdomains, V_1, V_2,..., V_k, reflects the assignment of cells (or BSs) to PCUs. Such a partition defines a set of edges that join vertices assigned to different subdomains, $\delta(V_1, ..., V_k)$. The sum of the weight of these edges, referred to as *edge-cut*, represents

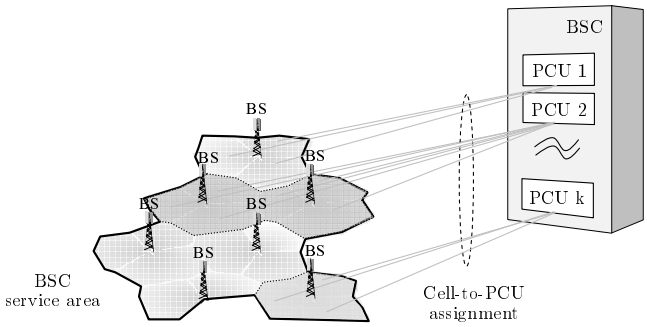


Fig. 1 The cell-to-PCU assignment problem.

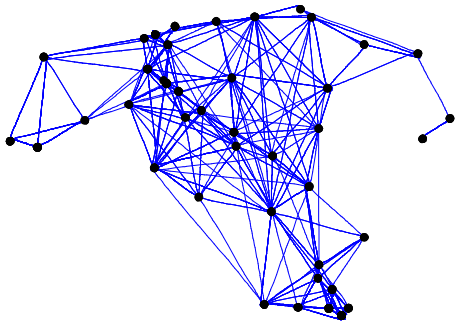


Fig. 2 Graph of a real base station controller.

the number of users that, after a change of BS, change their PCU. The *BS-to-PCU Assignment Problem* (BPAP) can be formulated as:

$$\text{Min} \quad \sum_{(i,j)\in\delta(V_1,\ldots,V_k)} \gamma_{ij} \tag{1}$$

$$\text{s.t.} \quad \|V_n\| = \sum_{i\in V_n} \omega_i \leq B_{aw}, \quad \forall\, n = 1:k, \tag{2}$$

$$\frac{\max(\|V_1\|,\ldots,\|V_k\|)}{\min(\|V_1\|,\ldots,\|V_k\|)} \leq B_{rw}\,, \tag{3}$$

where (1) reflects the goal of minimizing the number of users that change their PCU, (2) reflects the PCU capacity limit, B_{aw}, and (3) ensures that the load is evenly distributed among PCUs by limiting the maximum weight imbalance ratio, B_{rw} (i.e., the weight ratio of the largest to the smallest subdomain). Although there are other ways to formulate the load imbalance constraint, (3) is the one used in the industry.

In Fig. 2, it is observed that BPAP graphs are not planar and can be highly heterogeneous, unlike the graphs from meshes in the supercomputing area, which have frequently been used to develop and test partitioning algorithms. Another distinctive feature, not shown in the figure, is the heterogeneity of vertex weights, which makes balancing of subdomain weights more complicated.

The BPAP, although approached in this paper as a graph partitioning problem, could also be considered as a special case of the Capacitated Task Allocation Problem

(CTAP). Recent examples of exact and heuristic approaches to the CTAP can be found in the literature, although authors tend to focus on much smaller problem instances than occur in mobile networks. For example, Ernst et al (2006) consider exact solutions to a benchmark set of problem instances ranging from 15 to 41 tasks assigned to between 4 and 12 processors, whilst Lusa and Potts (2008) introduce a variable neighbourhood search heuristic for the same benchmark instances. In contrast, the problem instances considered here, taken from a real network, range from 85 to 213 base stations (tasks) assigned to between 4 and 8 controllers (processors).

3 Solution Technique

This section describes the proposed graph partitioning method. For clarity, the classical algorithms combined in the method are first introduced.

3.1 Classical Multi-Level Graph Partitioning Algorithm

Multi-Level (ML) schemes first coarsen 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, a local refinement algorithm is applied to small portions of the graph close to the partition boundary. This technique dramatically improves on single-level local search techniques in terms of solution quality and compares favorably in terms of runtime.

During the coarsening stage, a sequence of smaller graphs is constructed from the original graph. In most coarsening schemes, pairs of vertices are collapsed to form single vertices on the next level coarser graph. In this work, a matching is defined by selecting the heaviest edges in the graph in a greedy fashion, (Gupta, 1997), which is hereafter referred to as *Sorted Heavy Edge Matching* (SHEM). By hiding the heaviest edges, a larger decrease of edge-cut is normally achieved when partitioning the coarsest graph.

To build an initial partition, the coarsening process might continue until the number of vertices in the coarsest graph is the same as the number of subdomains k. Alternatively, the coarsening phase might end when the number of vertices falls below a certain threshold (e.g., c times the number of subdomains). In this approach, the *Greedy Graph Growing Partitioning* (GGGP) algorithm, (Karypis and Kumar, 1998a), is often adopted to build the initial partition. In its original version, a bisection of a graph is built by growing a subdomain incrementally around an arbitrary seed vertex. At each step, unassigned vertices are ordered in terms of the weight of edges to vertices already in the growing region. In this work, the *k-way GGGP* algorithm, (Krishnan et al, 1999), has been used. In this algorithm, k initial seed vertices are chosen and neighboring vertices are alternately added to the subdomain with the smallest weight.

During the uncoarsening phase, the partition of the coarser graph is projected back to the original graph. As graphs get finer, the increasing number of degrees of freedom can be used to improve the partition. Several refinement schemes have been proposed. The *greedy refinement*, (Karypis and Kumar, 1998b), moves vertices from one subdomain to another to reduce the edge-cut without exceeding the allowed weight imbalance following a steepest descent approach. Alternatively, the *Kernighan-Lin* algorithm, (Kernighan and Lin, 1970), allows movements that temporarily degrade the

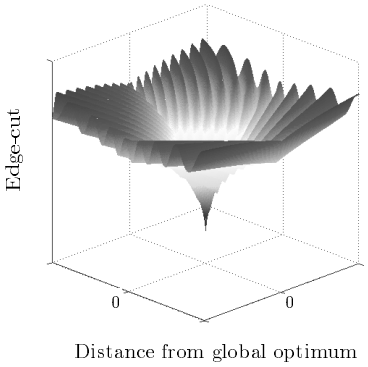


Fig. 3 A globally-convex optimization surface.

edge-cut to provide limited hill-climbing capability. Most graph partitioning packages implement the variant of this algorithm proposed by Fiduccia and Mattheyses (1982).

3.2 Classical Adaptive Multi-Start Algorithm

The quality of the solution obtained by local refinement is heavily influenced by that of the initial solution. Thus, a better solution can be found by performing several attempts, which is referred to as *multi-start* technique. By sampling the solution space, the diversity of the search is improved at the expense of a loss of intensity. In its simplest version, (Boender et al, 1982), the initial solutions are randomly selected, which in the graph partitioning case can be easily implemented by selecting seeds randomly in GGGP. This method is hereafter referred to as *Random GGGP* (R-GGGP). However, the effectiveness of this strategy is limited, since random local minima in large combinatorial optimization problems tend to all have intermediate quality with very little difference between them. As a consequence, the probability to improve a previous solution quickly diminishes from one attempt to the next.

The shortcomings of the previous approaches can be overcome by taking advantage of the regularity of the search space. In large combinatorial problems, the optimization surface tends to have a structure where the best local minima are grouped in a central position in the search space, (Boese et al, 1994). Such a surface with a "big central valley", as the one in Fig. 3, is said to be *globally-convex*. In the figure, it is seen that the optimization surface comprises multiple local minima, and hence the shortcomings of local search methods. However, the fact that the best local minima are close to each other can be used to improve the efficiency of the search.

The previous observation is the basis of *Adaptive Multi-Start* schemes. These methods exploit the regularity of the optimization surface to direct the search to regions where it is more likely that the best solution is found. This can be easily achieved by selecting starting points for local search algorithms from previously found local minima. Thus, the efficiency of the search is greatly improved, provided that regularities are correctly detected. In the graph partitioning problem, the regularity of the optimization surface is translated into similarities between the best partitions. Thus, the best partitions of a graph often share the same assignment of a large number of vertices.

Amongst all adaptive multi-start schemes for graph partitioning, the *Clustered Adaptive Multi-Start* (CAMS) algorithm, (Hagen and Kahng, 1997), is the most widely used scheme. In CAMS, new partitions are generated from the best partitions previously built. The core of the method is the detection of similarities among previous solutions, which is used to simplify the graph. Being a ML method itself, CAMS performs coarsening not only based on the graph structure, as traditional coarsening schemes do, but also on the similarities of previous solutions. Thus, coarsening is not performed statically, but dynamically, based on solutions built so far.

The CAMS algorithm starts by building an initial set of solutions by the naive multi-start approach (i.e., R-GGGP). On each subsequent iteration, the algorithm identifies clusters of vertices assigned to the same subdomain in all solutions. Once these clusters are identified, a simpler version of the graph is constructed by collapsing all vertices in a cluster into a single vertex. This matching operation is also performed over the existing set of solutions. A refinement algorithm is then applied over all existing solutions in the simplified version of graph. It is worth noting that, even though the matched solutions are essentially the same as their non-matched counterparts, the application of the refinement algorithm over a simpler graph does not necessarily lead to the same solutions. On the contrary, by collapsing new groups of vertices, the capability of local search algorithms to escape local minima is improved. As a result, a new set of solutions is obtained, over which the previous steps are repeated. As iterations pass, the graph is progressively simplified by detecting new similarities. This iterative process ends when all solutions coincide or the refinement process does not produce any change.

The main parameter in CAMS is the number of initial solutions, s, which controls the trade-off between diversity and intensity of the search. A large value of s ensures that most regions of the solution space are explored, increasing the diversity of the search. The main drawback is a reduced likelihood of a group of vertices being in the same subdomain for all solutions. As a result, a large number of iterations is needed to ensure that all solutions finally coincide, which might be considered as a loss of intensity in the search. In contrast, a small value of s ensures a fast convergence to the final solution at the expense of a higher variability of solution quality. Alternatively, the intensity of the search can also be controlled by limiting the number of generations obtained from the same set of initial solutions, g. In most cases, a small value of g reduces runtime at the expense of degrading solution quality.

3.3 Proposed Heuristic Algorithm

The proposed algorithm combines the multi-level and adaptive multi-start approaches into a single method. The template of the algorithm is shown in Fig. 4. As any ML scheme, the algorithm consists of three stages: coarsening, initial partitioning and uncoarsening. During coarsening, the original graph is simplified by SHEM. Over the coarsest graph, the initial partitioning is performed by CAMS. The subsequent uncoarsening stage projects the partition on the coarsest graph back to the original graph by unfolding matched vertices. During this stage, the classical FM refinement is applied after each uncoarsening operation. This method will hereafter be referred to as ML-CAMS algorithm.

Stage 1) **Coarsening stage**

1.1) **Repeat** Match vertices by *Sorted Heavy Edge Matching* algorithm

i. Rank edges based on decreasing weight by *Quicksort* algorithm

ii. Select next edge and match endvertices if not already matched, until no edge is left unchecked

until the average number of vertices per subdomain in the coarsest graph is below a certain threshold.

Stage 2) **Initial partitioning**

2.1) Build an initial set of s partitions of the coarsest graph by *Random Multi-Start* algorithm

i. Build the different partitions by *Greedy Graph Growing Partitioning* algorithm

ii. Refine each partition by *Fiduccia-Matheysses refinement* algorithm.

2.2) **Repeat** Generate a new set of g partitions from the old set of partitions

i. Find groups of vertices assigned to the same subdomain in the current set of partitions

ii. Simplify the coarsest graph by matching vertices in a group into a unique vertex and simplify the current set of partitions accordingly

iii. Refine each partition in the current partition set by *Fiduccia-Matheysses refinement* algorithm over the coarsest graph

iv. Build the new set of solutions by uncoarsening the new solutions from the previous step

until a maximum number of generations, g, have been reached or the new set of partitions coincide to the old one.

Stage 3) **Uncoarsening stage**

3.1) **Repeat** Progressively refine the initial partition on the coarsest graph

i. Uncoarsen the coarser graph based on the matching scheme

ii. Refine the current partition by *Fiduccia-Matheysses refinement* algorithm

until the finer graph is the original graph.

Fig. 4 Template of the proposed heuristic graph partitioning algorithm.

4 Analysis over Measurement-Based Network Model

To evaluate the performance of the different methods, an extensive set of graphs is built with measurements taken from a live mobile network. For clarity, the preliminary conditions are described first and the results of the analysis are subsequently presented.

4.1 Analysis Set-up

The network area optimized covers a geographical area of 150000 km^2, comprising 8952 BSs distributed over 61 BSCs. As the problem is solved on a per-BSC basis, the set of problem instances consists of 61 graphs. Table 1 presents some relevant statistics of the instances. The last row shows the mean and standard deviation of the log-normal distribution followed by edge weights.

The input data are the number of users moving between BSs during a 9-day period (i.e., γ_{ij}), the number of data channels per BS (i.e., w_i) and the number of PCUs per BSC (i.e., k). As optimization constraints, the maximum number of channels per PCU is 256 (i.e., $B_{aw} = 256$) and the maximum weight imbalance ratio is 2 (i.e., $B_{rw} = 2$).

The analysis first compares the performance of the methods over the whole set of 61 instances. During the analysis, four heuristics are compared: a) the naive multi-start

Table 1 Main statistics of the problem instances.

	Avg	Std	Min	Max
No. of vertices	146.8	27.5	85	213
No. of adjacencies	929.3	322.3	241	1907
Graph density [%]	4.4	1.2	1.5	7.7
No. of subdomains	5.3	1.0	4	8
Avg. no. of vertices per subdomain	27.8	4.5	15.9	37.4
Avg./Std. $\log_2$ of edge weight	6.6/2.0	0.3/0.1	5.2/1.6	7.3/2.3

algorithm, based on the repeated use of the GGGP algorithm with random seeds and greedy refinement (denoted as R-GGGP), b) the classical multi-level algorithm, with SHEM for coarsening until the number of vertices is the same as the number of subdomains and FM refinement during uncoarsening (denoted as ML), c) the clustered adaptive multi-start algorithm, using R-GGGP to build a set of initial solutions, which are checked for similarities to simplify the original graph iteratively and later improved by FM refinement (denoted as CAMS) and d) the proposed multi-level clustered adaptive multi-start algorithm, which coarsens the graph partially by SHEM and then uses CAMS only for the initial partitioning (denoted as ML-CAMS). To quantify the performance difference between these methods and the optimal solution, an integer linear programming model of the problem is built and solved per BSC (denoted as ILP). The exact ILP model, described in (Toril et al, 2010), improves that in (Merchant and Sengupta, 1995) by adding constraints to eliminate model symmetry, (Sherali and Smith, 2001). The 61 models are solved by SCIP v1.10/SoPlex v1.32.

All heuristic methods share the classical FM refinement algorithm, except R-GGGP, which uses the greedy variant. Unless stated otherwise, the number of passes in FM is 4. For a fair comparison with the exact solution, the refinement algorithm in this work does not force that subdomains are connected[1], as such a constraint cannot be included in the integer programming model. It is also worth noting that the implemented version of ML is deterministic, whereas CAMS, ML-CAMS and R-GGGP are randomized (and hence produce a different solution for each different random seed). To ensure the statistical confidence of results, performance figures for CAMS and ML-CAMS are computed from the average of 100 independent runs, while figures for R-GGGP correspond to 100 independent runs of the algorithm (i.e., the edge-cut of the best attempt and the runtime of the whole series of attempts).

Several performance indicators are evaluated during the assessment process. From the operator perspective, the main figure of merit is the edge-cut normalized by the sum of edge weight, which is hereafter referred to as *edge-cut ratio*. The execution time is also evaluated. For that purpose, the different routines are run on a Windows-based computer with a clock frequency of 2.8GHz and 1GByte of RAM. To assess the value of a given algorithm, the trade-off between runtime and solution quality is evaluated as in (Walshaw, 2004). Most optimization algorithms contain a parameter, referred to as intensity, that allows the user to specify how long the search for an optimal solution should continue before giving up. In this work, the intensity of ML is controlled by the number of passes in the FM algorithm, whereas the intensity of CAMS and R-GGGP is controlled by the number of generations and attempts, respectively. In ML-CAMS, the number of passes in FM and the number of generations in CAMS were

[1] A subdomain is *connected* if there is an internal path between all vertices in the subdomain.

Table 2 Overall performance of graph partitioning methods.

Heuristic method	ML	CAMS	ML-CAMS	R-GGGP	ILP
Sum of edge weights [$\cdot 10^6$]			243.7		
Total edge-cut [$\cdot 10^6$]	18.44	17.15	17.11	17.99	14.52
Edge-cut ratio [%]	7.57	7.04	7.02	7.38	5.96
Normalized edge-cut	1.27	1.18	1.18	1.24	1
Average weight imbalance ratio	1.93	1.92	1.94	1.87	1.99
Total runtime [s]	21	159	73	842	$2.5 \cdot 10^6$

simultaneously adjusted. For each intensity value and problem instance, edge-cut and runtime in different runs of the algorithm are averaged. The sum of the edge-cut and runtime averages obtained on a per-instance basis gives the total edge-cut and runtime in the network area.

A more detailed analysis of ML-CAMS is then performed over isolated problem instances. The analysis first shows several properties of the optimization surface of the problem that are the origin of the superior performance of adaptive multi-start approaches. For that purpose, a set of local minima is built by R-GGGP in a few instances selected at random. Finally, a sensitivity analysis of ML-CAMS is performed in a single problem instance.

4.2 Analysis Results

4.2.1 Performance Comparison over Aggregated Case

Table 2 presents the overall performance of the different methods. The performance of the heuristic methods is shown from the 2^{nd} to the 5^{th} column and that of the optimal solution is shown in the last column. The 2^{nd} row reflects the sum of edge weight in the area, corresponding to the edge-cut when every base station has its own PCU (i.e., worst-case). The 3^{rd} row reflects the quotient between the total edge-cut and the sum of edge weight (i.e., the edge-cut ratio). The 5^{th} row represents the edge-cut normalized by that of the optimal solution (i.e., normalized edge-cut).

In the table, it is observed that the classical ML algorithm achieves good solutions quickly. More specifically, ML finds a solution with total edge-cut 27% larger than that of the optimal solution in seconds. Thus, ML is the best option when runtime is the main concern. On the other extreme, R-GGGP only reduces the total edge-cut of ML by 2.4% at the expense of increasing runtime by a factor of 40. Note that, in spite of the large number of attempts (i.e., 100), the total edge-cut in R-GGGP is still 24% larger than the optimal value. By using adaptive multi-start, CAMS obtains a larger edge-cut reduction in less than 3 minutes. By coarsening the graph before applying CAMS, ML-CAMS halves the runtime without increasing the total edge-cut.

Fig. 5 shows the trade-off between edge-cut and runtime in the heuristic methods. To aid comparison, edge-cut values are normalized by the edge-cut of the optimal solution (i.e., ILP) and runtime values are normalized by the runtime of the fastest method (i.e., ML without local refinement). Fig. 5 shows that ML gives the fastest convergence to its best solution. In contrast, R-GGGP slowly converges to its best solution, stagnating after a few attempts. In CAMS curve, the 1^{st} point coincides with the 5^{th} point in R-GGGP curve, as the first generation of solutions in CAMS consists

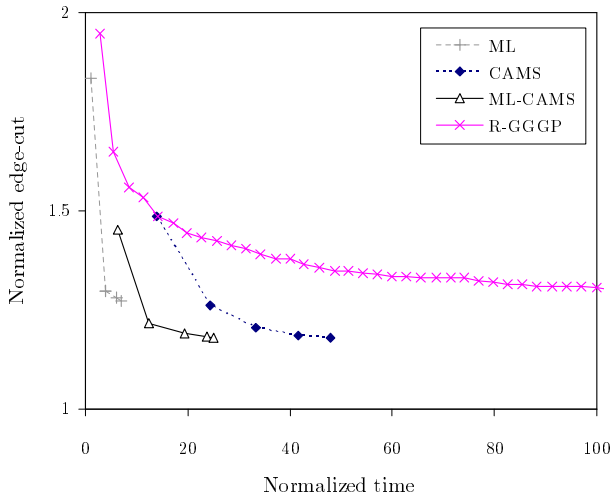


Fig. 5 Convergence behavior of heuristic methods.

of 5 solutions built by R-GGGP. Thereafter, the identification of similarities in CAMS greatly enhances the convergence speed. Finally, it is observed that ML-CAMS brings forward the benefits from adaptive multi-start techniques, giving quick solutions whose quality is close to those achieved by R-GGGP in the limit.

Once it has been shown that ML-CAMS performs extremely well over this type of graphs, the following analysis intends to justify the superiority of ML-CAMS by studying the problem optimization surface.

4.2.2 Analysis of Optimization Surface in a Test Case

The following analysis identifies a statistical model for the values of random local minima and proves the correlation among the best solutions in the solution space. For this purpose, a set of 1000 random local minima is computed on several instances of the problem by R-GGGP. This set of solutions is expected to be representative of the optimization surface, provided that seed vertices in GGGP are selected randomly.

Fig. 6 shows the histogram of edge-cut values of the local minima achieved by R-GGGP in a real problem instance. It is observed that the histogram exhibits a fairly regular structure, but it is clearly asymmetrical, with a tail extending to the right. This suggests that the extreme value distribution might be a distributional model for the local minimum values. In particular, the *extreme value distribution - type I* (also known as *Gumbel distribution*) proves to be an adequate model for the edge-cut values. The probability density function (PDF) of the Gumbel distribution (maximum) is

$$f(x;a,b) = \frac{1}{b}\exp\left[-\frac{x-a}{b} - \exp\left\{-\frac{x-a}{b}\right\}\right] \quad -\infty < x < \infty, \tag{4}$$

where a is the location parameter and b is the scale parameter. The corresponding cumulative density function (CDF) is

$$F(x;a,b) = \exp\left[-\exp\left\{-\frac{x-a}{b}\right\}\right] \quad -\infty < x < \infty. \tag{5}$$

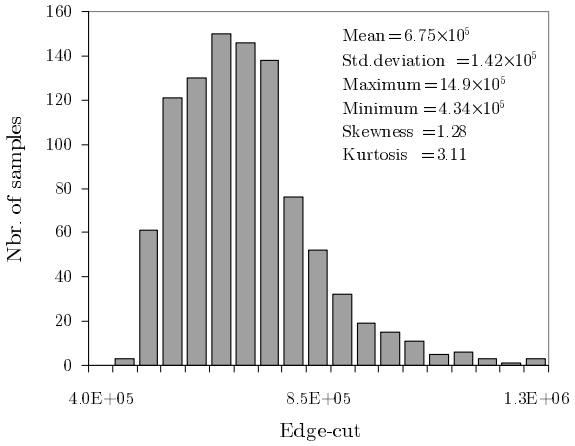


Fig. 6 Histogram of values of local minima from R-GGGP algorithm.

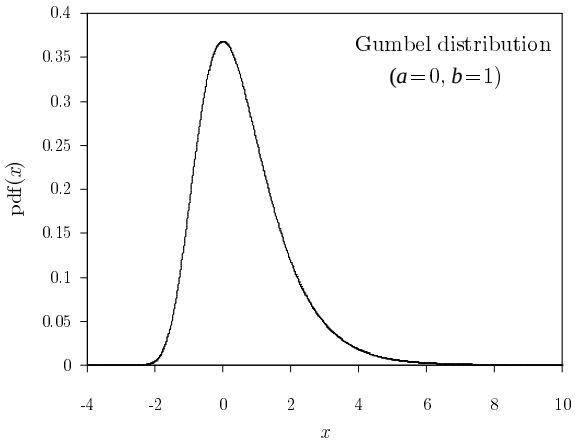


Fig. 7 Probability density function of the standard Gumbel distribution.

Fig. 7 depicts the PDF of the standard Gumbel distribution, where $a = 0$ and $b = 1$.

To assess how well the distributional model fits the data set, the CDF of both the sample data and the theoretical model are compared. For this purpose, the empirical cumulative distribution function (ECDF) of the observed random variable X (i.e., the edge-cut) is computed. Comparison of ECDF and CDF values is then performed by means of a probability plot. Fig. 8 presents the Gumbel probability plot of the local minima values in 3 problem instances. The linear relationship between both axis indicates clearly that the selected distribution does in fact fit the data very well in all instances. To reinforce this statement, the values of the squared sample correlation coefficient, R^2, are superimposed on the figure. The values of R^2 close to 1 give evidence of the strong correlation. The same trend is observed in the remaining 58 instances. More precisely, the average and minimum value of R^2 in the 61 instances are 0.980 and 0.903, respectively. From this result, it can be concluded that the local minimum values in any problem instance follow a Gumbel distribution. From Fig. 8, it can also be deduced that, although the distributional model remains valid for all instances,

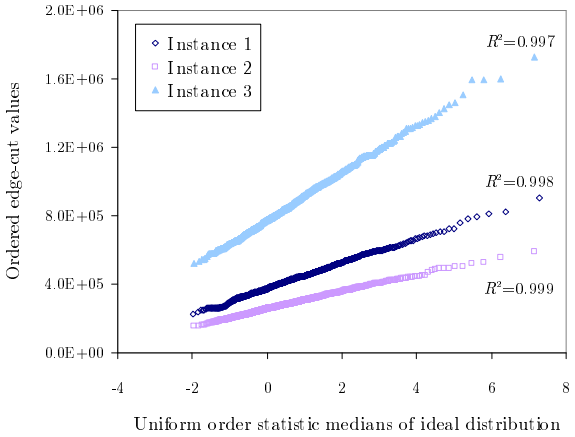


Fig. 8 Probability plot of values of local minima values in several problem instances.

the parameters in the model depend on each specific instance (otherwise, the curves of different instances would coincide). Hence, the parameters of the distribution must still be estimated on a per-instance basis.

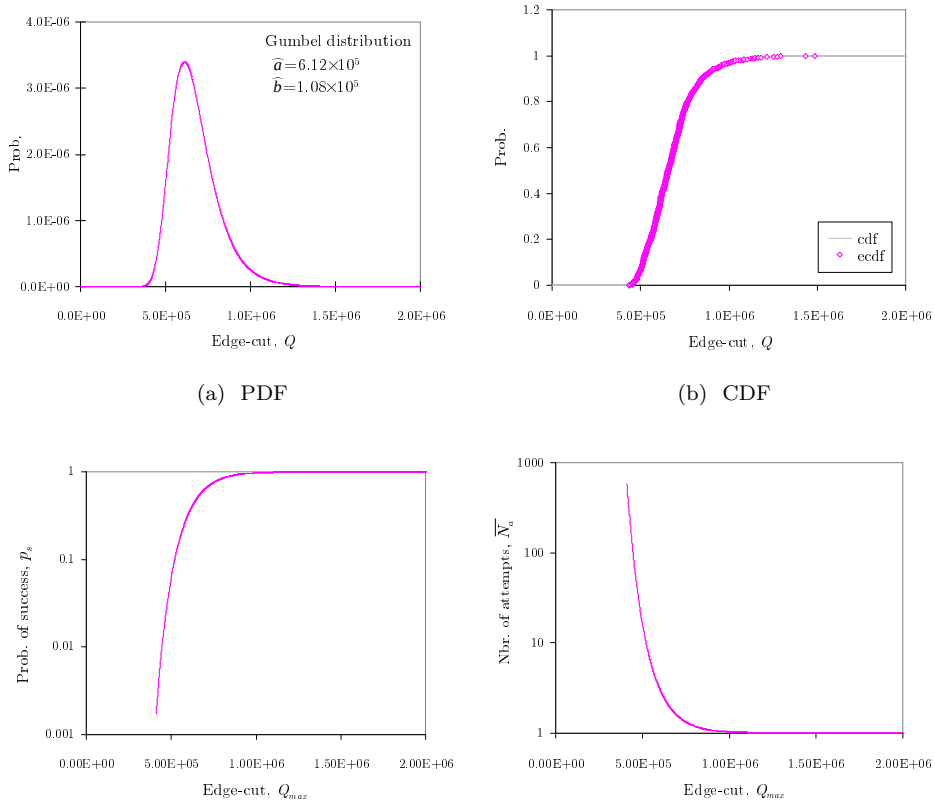
The analytical tool described so far can be used to show how the probability of improving a previous solution quickly diminishes from one attempt to the next in the naive MS approach. Thus, the number of R-GGGP attempts that must be performed to obtain a solution with a certain minimum quality (i.e., maximum edge-cut) is a random variable geometrically distributed. From (5), it is deduced that the probability of success on each individual attempt is

$$p_s = F(Q_{max}; a, b) = P(x \leq Q_{max}) = \exp\left[-\exp\left\{-\frac{Q_{max} - a}{b}\right\}\right], \qquad (6)$$

where Q_{max} is the target edge-cut value to be achieved. The average number of attempts until success, N_a, is equal to $\frac{1}{p_s}$.

Fig. 9 shows how the previous formulas can be used to estimate the number of R-GGGP attempts to reach a solution with a pre-defined edge-cut. The example is based on the same set of local minima considered in Fig. 6. First, the edge-cut PDF is modeled by a Gumbel distribution. For this purpose, maximum likelihood estimates of the distribution parameters, $\widehat{a}$ and $\widehat{b}$, are computed from the sample. Fig. 9 (a) presents the resulting PDF. Fig. 9 (b) shows the empirical and theoretical CDFs, where it is observed that both fully coincide. Fig. 9 (c) shows the success probability in a single attempt, p_s, as a function of the desired edge-cut value, Q_{max}. The rapid decay of the PDF below the location parameter leads to a fast decrease of p_s for small values of Q_{max}. As a consequence, the average number of attempts increases exponentially as Q_{max} decreases below the location parameter, as shown in Fig. 9 (d). This behavior explains the stagnation of naive MS approaches after a few attempts.

The preliminary analysis concludes with the proof of the correlation among the best solutions in the search space. This property is verified over the previous set of 1000 random local minima built for a problem instance. The scatter plot in Fig. 10 (a) shows the relationship between edge-cut and average distance to all other local minima in the sample. The distance between two solutions is defined as the number of vertices not assigned to the same subdomain in both solutions. In the figure, it is



(a) PDF (b) CDF

(c) Success probability (d) Avg. number of attempts

Fig. 9 Example of performance analysis of the R-GGGP method.

observed that the best local minima (i.e., the ones with the lowest edge-cut) have the smallest average distance to other minima. It can thus be concluded that the best local minima are central to all other local minima, since other minima are located around them. In this example, the best local minimum is exactly in the centre, since it has the smallest average distance to the other minima. Fig. 10 (b) plots edge-cut versus distance to the best local minimum. It is observed that there exists a clear correlation between solution quality and distance to the best local minima. All these facts suggest a globally-convex structure for the optimization surface, where the best local minima are grouped in the middle of the solution space. This result indicates that the best solutions share common components, which is the basis of adaptive MS approaches.

The regularity of the optimization surface is increased by the smoothing effect of the coarsening process, (Walshaw and Everett, 2002). Fig. 11 presents the analysis of 1000 random local minima obtained by R-GGGP over a coarsened version of the original graph. Again, it is observed that the best local minima show smaller average distance to other local minima. Thus, the best local minima tend to be central to all other local minima. However, in the original graph, the best 20 local minima are not fully clustered. On the contrary, Fig. 10 (a) shows that one local minimum with

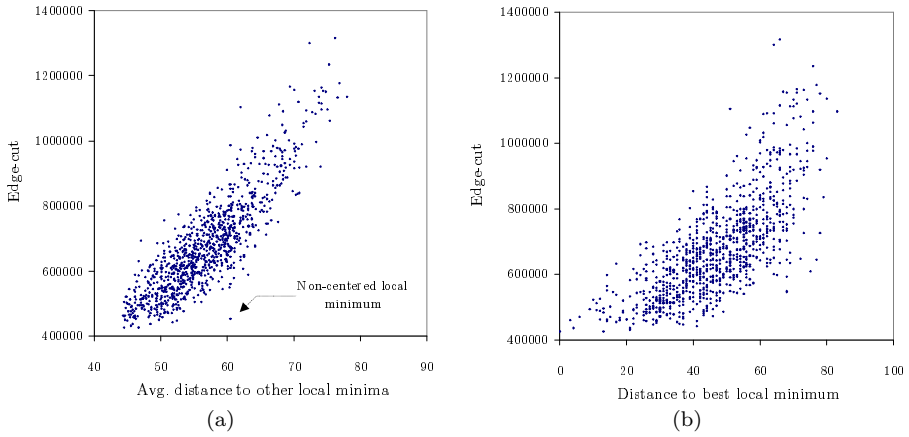


Fig. 10 Correlation between distance and edge-cut in local minima in the original graph.

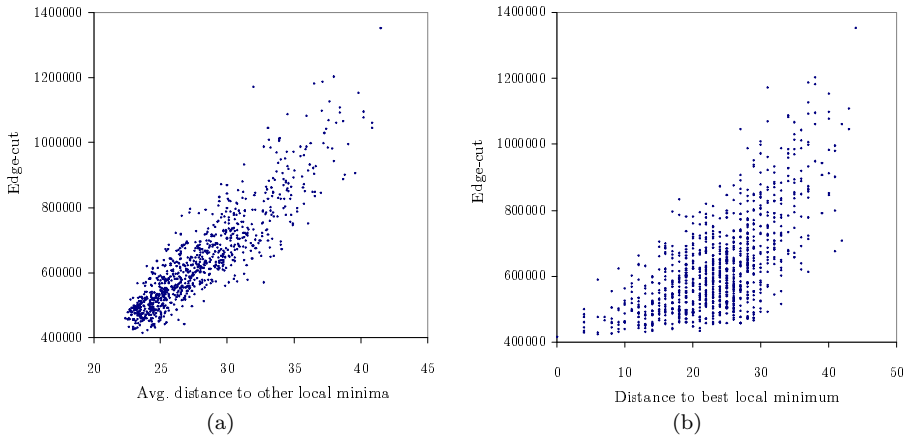


Fig. 11 Correlation between distance and edge-cut in local minima in the coarsened graph.

edge-cut close to the minimum value has an average distance of 60, which is much larger than the minimum one of 45. In contrast, the best local minima in Fig. 11 (a) are better grouped, which is evident from the narrower scatter plot. These figures also show that, after coarsening, most random local minima are clustered in the low edge-cut region. By hiding edges with large weights, coarsening the graph filters out the worst local minima, which improves the quality of R-GGGP solutions. In parallel, coarsening reduces the distance between local minima, due to the reduction of the number of vertices in the graph (note the different x-axis scale in Fig. 10 and Fig. 11). The smaller distance between solutions leads to a more efficient refinement process, which is key to the runtime efficiency of CAMS. All these facts justify the superiority of ML-CAMS over CAMS both in terms of solution quality and runtime.

Fig. 11 (a) also explains a more subtle property of adaptive multi-start methods. In the figure, it is shown that the best local minimum is not necessarily in the centre of the solution space. In this case, the best minimum is not the one with the minimum average distance to other local minima (i.e., there exist other minima to the left of

the lowest one in the figure). This result suggests that the best minimum is displaced from the centre of gravity of the set of local minima (otherwise, it would display the minimum average distance to all other minima). Thus, an algorithm that built a new partition by minimizing the average distance to a previous solution set would fail to find the best solution if the latter was not completely centered. In contrast, CAMS does not intend to minimize the average distance to previous solutions, but is only guided by similarities among solutions. Thus, new solutions built by CAMS from an old set of solutions have lower average distance to the old solutions (i.e., they are interior points of the old set), but do not necessarily have the minimum distance (i.e., they might not be in the geometrical centre).

4.2.3 Sensitivity analysis of ML-CAMS in a Test Case

Finally, the analysis evaluates the sensitivity of ML-CAMS to changes in the parameters controlling the intensity and diversity of the search: the maximum number of generations, g, and the number of solutions per generation, s. The aim is to find the best settings for these parameters.

To gain some insight into the influence of the number of generations on solution quality, the first experiment shows how solution quality is improved across generations. For this purpose, 1000 ML-CAMS attempts are performed on a single problem instance and the quality of solutions in each generation is evaluated. Initially, no restriction is imposed on the number of generations (i.e., g is set to ∞). Fig. 12 illustrates the ECDF of edge-cut in the solutions of each generation. In the figure, it is observed that most of the edge-cut improvement takes place in the construction of the 2^{nd} generation of solutions (i.e., 1^{st} generation after detecting similarities). Subsequent generations only provide marginal improvement, mainly from the elimination of the worst-quality individuals in the solution set (i.e., solutions to the right of the x-axis). This result proves that a small benefit is obtained after the 2^{nd} generation. Although it might be tempting to limit the maximum number of generations to increase the efficiency of the method, Fig. 13 proves that it is not necessary. Fig. 13 depicts the histogram of the number of generations in the 1000 ML-CAMS attempts. From the frequency values, it is observed that 650 out of 1000 attempts only had one generation in addition to the initial one built by R-GGGP. From the ECDF values in the secondary axis, it is also evident that 95% of the attempts had 3 or less generations. The reason for this early stop is the inability to generate a different solution set after 2-3 generations, since no additional similarities in the solution set tend to appear beyond that point.

Finally, the analysis evaluates the influence of the number of solutions per generation on solution quality. For this purpose, 1000 independent runs of ML-CAMS are carried out for different values of this parameter. Fig. 14 depicts a box plot to show the effect of this parameter on edge-cut performance. The box stretches between the 25% and 75% tiles. The median is shown as a dotted line across the box, while the mean and extreme values are represented by symbols. A trend line has also been superimposed to highlight the influence of s on the average edge-cut performance. In the figure, it is observed that the performance of the different parameter settings mainly differ in the higher edge-cut region, especially in the maximum (i.e., worst-case) value. Results show that ML-CAMS exhibits a large edge-cut variability for $s = 3$, which suggests that three solutions per generation do not provide enough diversity. In contrast, setting $s = 5$ largely reduces the worst-case value without an excessive increase in runtime. Beyond that value, the benefit from an increased diversity is significantly reduced and

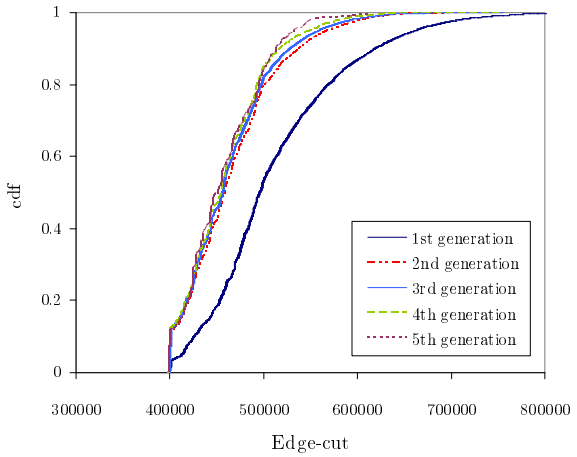


Fig. 12 Evolution of edge-cut across generations in ML-CAMS.

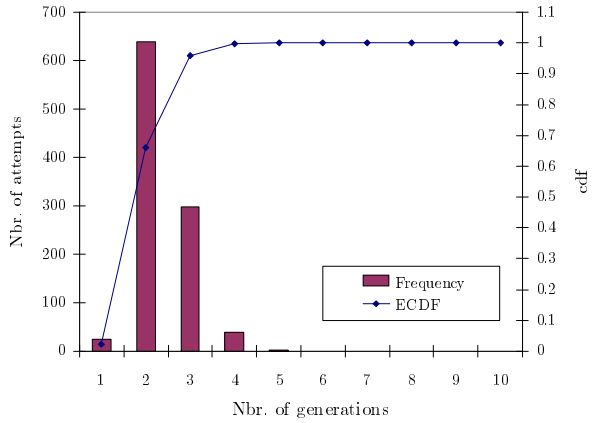


Fig. 13 Histogram of the number of generations in ML-CAMS.

does not compensate for the increased runtime. Although it might be expected that runtime grew linear in s, experiments prove that the growth is significantly larger due to an increase of the number of generations in the algorithm. The histogram presented in Fig. 15 shows that the number of generations increases with the number of individuals per generation. This result is mainly due to the difficulty of finding similarities in a large set of solutions. As more solutions are refined during an increased number of generations, the convergence to the final solution is much slower.

5 Conclusions

In this paper, a detailed analysis of a graph partitioning algorithm for assigning base stations to packet control units in a mobile network has been presented. The proposed algorithm first coarsens the original graph by sorted heavy edge matching, then performs an initial partitioning by the clustered adaptive multi-start algorithm and later

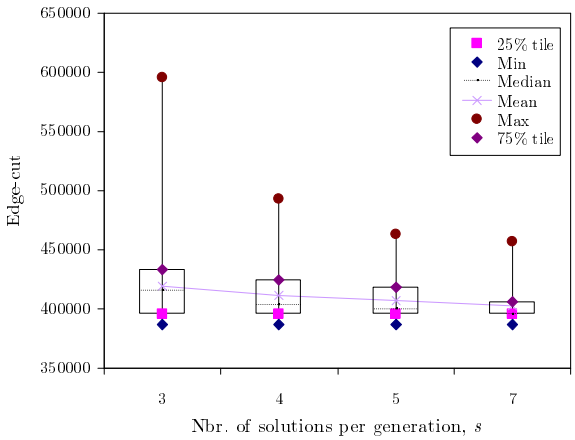


Fig. 14 A box plot of the edge-cut distribution for different number of solutions per generation in ML-CAMS.

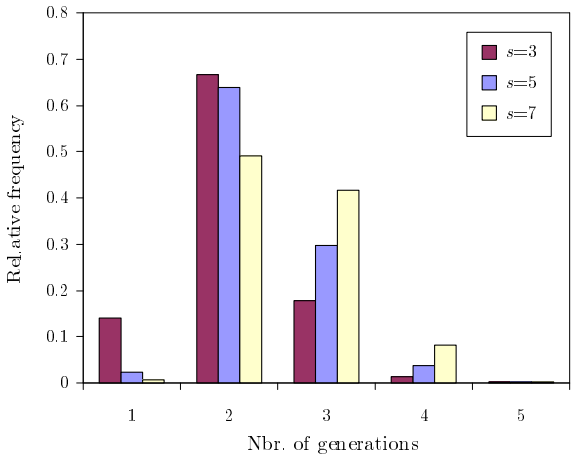


Fig. 15 Histogram of the number of generations for different number of solutions per generation in ML-CAMS.

unfolds the resulting partition applying the Fiduccia-Mattheyses refinement algorithm. Performance assessment has been performed over an extensive set of problem instances built with measurements from a live mobile network. A preliminary analysis of the optimization surface has shown that, even though the problem has lots of local minima, the best local minima bear strong similarities. Such similarities are increased when coarsening the graph. This is the reason for the benefit of combining multi-level and adaptive multi-start techniques. Overall results have confirmed that the proposed algorithm can decrease the total edge-cut obtained by the classical multi-level algorithm by 7% without increasing runtime as much as the classical adaptive multi-start algorithm. The trade-off between solution quality and efficiency in the method is controlled by a proper selection of the number of solutions per generation in the clustered adaptive multi-start algorithm. Nonetheless, the performance gap with the optimal solution indicates that there is still some room for improvement. Experiments reported in (Toril,

2007) show that other time-efficient heuristics obtain much larger differences. This is a clear indication of the complexity of the problem, mainly due to the small number of vertices per subdomain and the heterogeneity of vertex weights, which makes load balancing difficult.

Although the proposed method has been conceived for a specific clustering problem in GERAN, it is equally applicable to similar clustering problems in other layers of network hierarchy (e.g., assignment of base station controllers and location areas) and other radio access technologies (e.g., third and fourth generation mobile networks).

References

Bhattacharjee PS, Saha D, Mukherjee A (2004) An approach for location area planning in a personal communication services network (PCSN). IEEE Transactions on Wireless Communications 3(4):1176–1187

Boender CGE, Rinnooy Kan AHG, Timmer GT, Stougie L (1982) A stochastic method for global optimization. Mathematical Programmming 22:125–140

Boese KD, Khang A, Muddu S (1994) A new adaptive multi-start technique for combinatorial global optimizations. Operation Research Letters 16:101–113

Demirkol I, Ersoy C, Caglayan MU, Delic H (2004) Location area planning and cell-to-switch assignment in cellular networks. IEEE Transactions on Wireless Communications 3(3):880–890

Donath WE, Hoffman AJ (1973) Lower bounds for the partitioning of graphs. IBM Journal of Research and Development 17(5):420–425

Ernst A, Jiang H, Krishnamoorthy M (2006) Exact solutions to task allocation problems. Management Science 52(10):1634–1646

Ferreira CE, Martin A, de Souza CC, Weismantel R, Wolsey LA (1998) The node capacitated graph partitioning problem: A computational study. Mathematical Programming 81(2):229–256

Fiduccia C, Mattheyses R (1982) A linear time heuristic for improving network partitions. In: Proc. 19th ACM/IEEE Design Automation Conference, pp 175–181

Gondim P (1996) Genetic algorithms and location area partitioning problem in cellular networks. In: Proc. 46th IEEE Vehicular Technology Conference, pp 1835–1838

Gupta A (1997) Fast and effective algorithms for graph partitioning and sparse matrix ordering. IBM Journal of Research and Development 41(1/2):171–184

Hagen L, Kahng A (1997) Combining problem reduction and adaptive multi-start: A new technique for superior iterative partitioning. IEEE Transactions On Computer-Aided Design of Integrated Circuits and Systems 16(7):709–717

Hendrickson B, Leland R (1995) A multilevel algorithm for partitioning graphs. In: Proc. 1995 ACM/IEEE Conference on Supercomputing, ACM Press

Karypis G, Kumar V (1998a) A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing 20(1):359–392

Karypis G, Kumar V (1998b) Multilevel k-way partitioning scheme for irregular graphs. Journal of Parallel and Distributed Computing 48(1):96–129

Kernighan BW, Lin S (1970) An efficient heuristic procedure for partitioning graphs. Bell System Technical Journal 49:291–307

Krishnan R, Ramanathan R, Steentrup M (1999) Optimization algorithms for large self-structuring networks. In: Proc. INFOCOM '99, vol 1, pp 71–78

Laiho J, Wacker A, Novosad T (2002) Radio Network Planning and Optimisation for UMTS. John Wiley & Sons

Lisser A, Rendl F (2003) Graph partitioning using linear and semidefinite programming. Mathematical Programming 95(1):91–101

Lusa A, Potts C (2008) A variable neighbourhood search algorithm for the constrained task allocation problem. Journal of the Operational Research Society 59(6):812–822

Merchant A, Sengupta B (1995) Assignment of cells to switches in PCS networks. IEEE/ACM Transactions on Networking 3(5):521–526

Mishra AR (2007) Advanced cellular network planning and optimisation. John Wiley & Sons

Mouly M, Pautet MB (1992) The GSM system for mobile communications. Cell & Sys

Pierre S, Houeto F (2002) A tabu-search approach for assigning cells to switches in cellular mobile networks. Computer Communications 25(5):465–478

Plehn J (1995) The design of location areas in a GSM-network. In: Proc. 45th IEEE Vehicular Technology Conference, pp 871–875

Rolland E, Pirkul H, Glover F (1996) Tabu search for graph partitioning. Annals of Operational Research 63:209–232

Saha D, Mukherjee A, Bhattacharjee PS (2000) A simple heuristic for assignment of cells to switches in a PCS network. Wireless Personal Communications 12:209–224

Schloegel K, Karypis G, Kumar V (2000) Graph partitioning for high performance scientific simulations. In: Dongarra J, Foster I, Fox G, Kennedy K, White A (eds) CRPC Parallel Computing Handbook, Morgan Kaufmann

Sherali HD, Smith JC (2001) Improving discrete model representations via symmetry considerations. Management Science 47(10):1396–1407

Toril M (2007) Self-tuning algorithms for the assignment of packet control units and handover parameters in GERAN. PhD thesis, University of Málaga

Toril M, Wille V, Barco R (2006) Optimization of the assignment of cells to packet control units in GERAN. IEEE Communications Letters 10(3):219–221

Toril M, Wille V, Molina-Fernández I, Walshaw C (2010) An adaptive multi-start graph partitioning algorithm for structuring cellular networks. Tech. Rep. IC-10-01, University of Málaga, URL http://mobilenet.ic.uma.es/images/stories/Publications/IC-10-01.pdf

Walshaw C (2004) Multilevel refinement for combinatorial optimisation problems. Annals of Operations Research 131:325–372

Walshaw C, Cross M (2000) Mesh partitioning: a multilevel balancing and refinement algorithm. SIAM Journal of Scientific Computing 22(1):63–80

Walshaw C, Everett MG (2002) Multilevel Landscapes in Combinatorial Optimisation; Tech. Rep. 02/IM/93. Tech. rep.