Comparison of heuristic approaches to PCI planning for Quantum Computers

Comparison of heuristic approaches to PCI planning for Quantum Computers

Giuseppe Barillaro*, Andrea Boella*, Filippo Gandino‡, Mohammad Ghazi Vakili¶,
Edoardo Giusto‡, Giovanni Mondo†, Bartolomeo Montrucchio‡,
Andrea Scarabosio §, Alberto Scionti §, Olivier Terzo §, Giacomo Vitali §
*Service Innovation, TIM S.p.A., Torino, Italy
†Strategy & Business Development, TIM S.p.A., Torino, Italy
‡DAUIN, Politecnico di Torino, Torino, Italy
§Advanced Computing and Applications, LINKS Foundation, Torino, Italy
¶University of Toronto, Canada
Email: andrea.scarabosio@linksfoundation.com, edoardo.giusto@polito.it

***Abstract*—Quantum Computing (QC) provides the possibility to develop new approaches to tackle complex problems. Real-world applications, however, cannot yet be managed directly due to the limitation of present and near-future noisy intermediate-scale quantum (NISQ) computers. Decomposition into smaller and manageable subproblems is often needed to take advantage of QC even when using hybrid (classical-quantum) solvers or solvers that already apply decomposition techniques. In this paper, heuristic decomposition algorithms to solve the Physical Cell Identifier (PCI) problem in 4G cellular networks in a way suitable for QC are presented. The PCI problem can be viewed as a map coloring problem with additional constraints and has been represented in a Quadratic Unconstrained Binary Optimization (QUBO) model, a form that, for instance, a quantum annealing machine could crunch. We propose two strategies, with variable decomposition granularity. The first one solves the problem recursively through bisection (max-cut problem), to use only one qubit to represent the status of the objects, avoiding one-hot encoding and thus minimizing the qubit requirement. The second is a multi-step approach, finally solving sets of randomized modified max-k-cut problems of customizable qubit size. We executed the algorithms on real cellular networks of one of the main Italian national telecom operators (TIM). The results show that all proposed QUBO approaches can be effectively applied to very large problems with similar or better performance of the reference classical algorithm, paving the way for the use on NISQ computers.**

***Index Terms*—PCI, quantum computing, heuristic, LTE, 4G, TelCo**

I. Introduction

As the number of smartphones increases around the world, wireless communication networks become more massive and denser, with the consequence that providing good connectivity quality is not an easy task. Long Term Evolution (LTE/LTE-A) standard aims to tackle these issues, satisfying the data demands for User Equipment (UE). LTE is based on the Orthogonal Frequency Division Multiple Access (OFDMA) for downlink (DL), necessary to guarantee the reliable transmission of data to a UE anywhere inside the network [1], [2].

The OFDMA method has been applied to reduce inter-cell interference (ICI) and intersymbol interference (ISI) problems in the downlink network. Physical Cell Identifiers (PCI), if properly assigned, can help improve the quality of service, in particular, to manage the mobility procedures of the user equipment (UE) to move from one cell to an adjacent one [2], [3].

PCI planning aims to assign every cell in the network a pseudo-unique identification number, depending on a set of conditions which will be detailed later in the text. This problem is usually formulated as a graph coloring problem in the computer science domain.

The graph coloring problem belongs to the category of non-deterministic polynomial-time (NP-hard) optimization problems [3]. Such kinds of problems have been addressed by heuristic and metaheuristic approaches [4]–[6].

Several studies have analyzed the PCI planning problems with respect to collision and confusion between cells [7], [8]. Quantum Annealing (QA) is a metaheuristic technique which naturally (quantum-mechanically) looks for the global minimum to a given objective function [9].

The leading company in the QA domain is D-Wave. Their Quantum Processing Unit (QPU) is particularly suited to solve optimization problems that can be formulated in a Quadratic Unconstrained Binary Optimization (QUBO) form.

Regardless of problem formulation into QUBO form, there is the issue of fitting the problem size to a dimension that fits the number of qubits available nowadays on a D-wave quantum annealer.

Our contribution in this paper is to formulate the PCI problem into sets of QUBO problems (decomposition), which could potentially be solved more easily by current and near term quantum computer (NISQ) with limited number of qubits and limited connectivity. We explore the effect of the granularity of the sub-problem and the effectiveness of this approach on very large PCI networks.

The paper is organized as follows: Section II describes the main aspects of QUBO; the PCI planning problem and models are presented in Section III; the new quantum inspired algorithms are described in detail in IV; Section V deals with the comparison of the algorithms with respect to some performance indicators, benchmarking them with the existing

TIM's approach; finally, some concluding and future works remarks are presented in Section VI.

II. QUBO MODEL

Quantum machines can be used to find the ground state of a target objective function, which is expressed in the Quadratic Unconstrained Binary Optimization (QUBO) model.

This function is made of variables that represent the qubits of the QPU. The best solution for a problem is represented by the set of variables values that lead to the lowest possible value of the objective function, which corresponds to the least possible energy of the system [10] [11].

The QUBO model can be expressed in this way:

$$f(x) = \sum_i Q_{i,i} x_i + \sum_{i<j} Q_{i,j} x_i x_j \tag{1}$$

where Q is an upper triangular matrix of weights, and x is the array of binary variables to be assigned. The Q matrix is made by qubit biases coefficients as diagonal terms, and by qubit couplers as off-diagonal terms.

The following is an equivalent, more compact matrix QUBO formulation of the problem:

$$\min_{x \in \{0,1\}^n} x^T Q x \tag{2}$$

III. PCI PLANNING PROBLEM

This section describes the PCI problem, describing the situation faced in reality by TelCo operators and giving a mathematical representation of the model itself and its constraints.

A. PCI planning - Definition

In the telecommunication domain, a *cell* is a land area corresponding to one transceiver antenna and a specific frequency band. The identification of the cell, via the PCI planning, allows mobile devices to handle the so-called *handover* between cells which are geographically adjacent. The PCI number is assigned to each of the cells in a mobile network, and it is chosen inside a predefined set of values. As stated in [12], the PCI is thus defined:

$$PCI = 3 \times ID_{group} + ID_{cell} \tag{3}$$

where ID_{group} is a number characteristic of the site, and ID_{cell} is a number characteristic of the cell within the site. ID_{cell} and ID_{group} can be also described in the literature as *Primary Synchronization Signal (PSS)* and *Secondary Synchronization Signal (SSS)*. A site groups from 1 up to 3 cells. Values for ID_{group} and ID_{cell} for 4G need to be in the range $[0; 167]$ and $[0; 2]$ respectively, which results in the PCI number being in the range $[0; 503]$.

B. PCI planning - Modelling

This subsection describes the mathematical model adopted to represent the PCI problem. The cost representation is a straightforward and easy way of evaluating the quality of a PCI plan. These costs depend on both the geographical conformation and coverage of the territory and the importance given by the TelCo operator to the unfulfilled requirement based on the know-how and experience.

For the sake of simplicity, it is supposed that each site is composed by at most three cells. Taking two generic cells, i and j, we have two costs binding them:

- $C_{i,j}$: cost of assigning the two cells the same PCI number;
- $S_{i,j}$: cost of assigning the two cells the same ID_{group}.

The total cost to be minimised is thus defined:

$$Tot_{cost} = \sum_i \sum_j C_{i,j} * V_{i,j} + \sum_i \sum_j S_{i,j} * W_{i,j} \tag{4}$$

where:

$$V_{i,j} = \begin{cases} 1 & \text{if the same PCI is assigned to both cells} \\ 0 & \text{otherwise} \end{cases}$$

$$W_{i,j} = \begin{cases} 1 & \text{if the same } ID_{group} \text{ is assigned to both cells} \\ 0 & \text{otherwise} \end{cases}$$

For the sake of comparing the performance of the different algorithm, we tried to minimise the number of different ID_{group} satisfying the following constraints:

- each cell of the same site must have:
  - the same ID_{group},
  - a different ID_{Cell};
- the cost $\sum_i \sum_j C_{i,j} * V_{i,j}$ must be zero.

For the same amount of used ID_{group}, Tot_{cost} must be minimized.

IV. DESCRIPTION OF PCI SCHEDULING ALGORITHMS

In this sectio the QUBO-based different approaches to the PCI scheduling problem are defined.

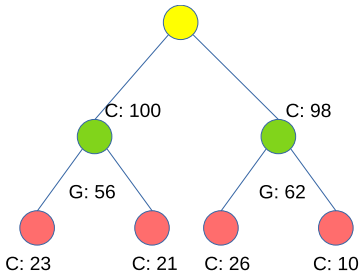
A. Single-Phase QUBO

A direct and straightforward implementation of the PCI problem in section III is possible in the following way: we assign simultaneously ID_{group} *and* ID_{Cell} using $k = 3 \cdot n_{ID_{group}}$ colors in a weighted max-k-cut problem (Single-Phase QUBO - SP). Then, for each cell, we assign $ID_{group} = floor(j/3)$ and $ID_{Cell} = mod(j, 3)$ where $j = 1, ..., k$ is the color assigned to the cell. How to formulate a weighted max-k-cut problem in QUBO format with one-hot bit encoding can be found in [10], [11]. As formulated so far, the problem does not guarantee that all cells of a single site (typically 3 cells) are assigned a different ID_{Cell} (see section III-B). This is an hard constraint and we need to add a penalty constraint when this situation is not satisfied in the solution. Using $\mathbf{X}$ as the binary solution matrix (of dimension $[\#cells; k]$) where $X_{i,j} = 1$ if the cell i is assigned the color j the extra linear constraints can be written as:

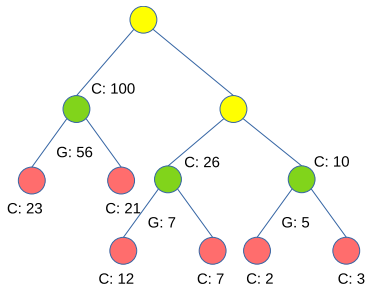
$$\Sigma_{k=1,3} \left(X_{i,p_j+k} - 2X_{i+1,p_j+k} + X_{i+2,p_j+k} \right) = 0 \tag{5}$$

$$p_j = j * 3, \, \forall i = 1, ..., \#sites, \, \forall j = 1, ..., floor(k/3) \tag{6}$$

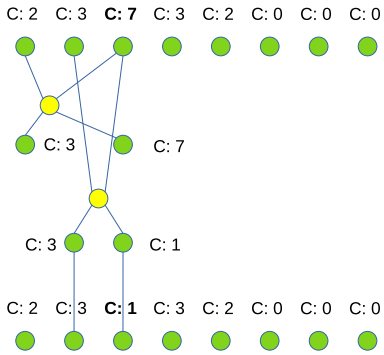
Combined with the standard single color occupancy requirements, Eq. 5 is satisfied only when the three cells of the same site i, $i + 1$ and $i + 2$ are assigned the same ID_{group}. The QUBO unknown vector x in eq. 2 is simply defined as the



(a) The tree that represents a step of dividing the sites among the ID_{Group} (green nodes).



(b) The tree of Fig.1a after a bisection.



(c) The Retroaction phase.

vectorisation of $\mathbf{X}$, $x = vec(\mathbf{X})$. General linear constraints in the form $\mathbf{A} \cdot x = b$ can be introduced in a QUBO model as additional quadratic penalties [10], [11].

B. Bisection Blossom approach

The main idea of the Bisection Blossom (BB), is to solve the problem through bisection, in order to use only one qubit to represent the status of one system variable avoiding *one-hot* encoding. This choice minimises the number of needed qubits but requires to solve a large number of QUBO problems. The ID_{Group} assignment is completed before performing the ID_{Cell} assignment, dividing the optimisation in two main phases.

1) ID_{Group} assignment: The goal of this task is to assign a ID_{Group} to any site, reducing the violations of the constraints described in Section III-B. To facilitate this assignment an extra cost is introduced representing the *Incompatibility* constraint.

a) Incompatibility: In addition to the costs $C_{i,j}$ and $S_{i,j}$, a third cost $F_{i,j}$ was introduced to represent the direct incompatibility between two sites to have the same ID_{group}. For each pair of sites, if there is no PCI combination such that the cells of the two sites can have the same ID_{group} with $C_{i,j} * V_{i,j} = 0$, then $F_{i,j} = 1$ for all cells of the two sites. Otherwise, $F_{i,j} = 0$.

The procedure begins by minimizing only the incompatibility. When $\sum_i \sum_j F_{i,j} = 0$, the procedure begins minimising $C_{i,j}$. At the beginning, all the groups have the same ID_{Group} and the cost can be represented by a graph with all sites as nodes and all costs (F or C) as edges. Then each bipartition splits the sites of one ID_{Group} into two groups with two different IDs_{Group}.

In this way, a binary tree is generated, in which the *root* is the original single ID_{Group} with all the sites, and the *leaves* are the selected groups of sites with a common ID_{Group}. The nodes between the *root* and the *leaves* are no more used, and are defined as *branch*. At each new level of the tree, the problem gets simpler, since the previous graph representing the cost has been split in two separate graphs, and each graph has about half of the nodes and about a quarter of the edges.

In order to decide which *leaf* must be split, a bipartition is applied on all the *leaves* and the new potential nodes are called *flowers*, which are at the moment only considered potential *leaves*.

This task is composed by a *setup* phase, a *loop* and a *retroaction*.

b) Setup: At the beginning, all the groups have the same ID_{Group}. Then sites are divided into two channels, minimizing the cost through the quantum annealer or a decomposing solver crunching a QUBO like `qbsolv`. The cost of both channels is calculated, and each new channel with a cost greater than 0 is divided into two potential channels. Finally, the cost of the *flowers* is calculated, and the gain due to a *leaf* "bloom" is calculated like the cost of the *leaf* minus the cost of the two *flowers*.

c) Loop: As long as the number of *leaves* is less than the number of ID_{Group} available and the sum of the cost of the *leaves* is greater than 0, the following steps are iterated: (i) the *leaf* with the greatest gain "blossoms", (ii) the *leaf* becomes *branch* (yellow), (iii) its *flowers* become *leaves* and generate four new *flowers*, (iv) new costs and gains are calculated.

d) Retroaction: Each group of sites with the same ID_{Group} that involves a cost $\sum_i \sum_j C_{i,j} * V_{i,j} > 5$ or $\sum_i \sum_j F_{i,j} > 0$ could not be solvable during the ID_{Cell} assignment, so is modified during the retroaction.

At the end of the *loop*, if there are too high costs, a *retroaction* of one level is applied. Each pair of *leaves* which include a leaf with a too high cost are merged and biparted. If the cost is decreased the new *leaves* are hold, otherwise the older ones. If at the end of the retroaction of level one some costs are still too high, a retroaction of level two is executed. In this case, four ID_{Group} are merged and then 3 bipartitions are applied. Finally, a retroaction of level one is performed in order to minimize Tot_{cost}.

2) ID_{Cell} assignment: Each site must assign a different ID_{Cell}, so each site has a maximum of 6 possible states:

(0,1,2) , (0,2,1) , (1,0,2) , (1,2,0) , (2,0,1) , (2,1,0)

The problem was solved using 6 qubits with *one-hot* encoding. A new cost matrix is calculated based on the six possible configurations of each site with the same ID_{Group}.

The problem is initially split in clusters of n sites in numerical order. This partition is applied only at the initial level of the tree, since at each level the number of sites decreases. The results presented here were obtained using

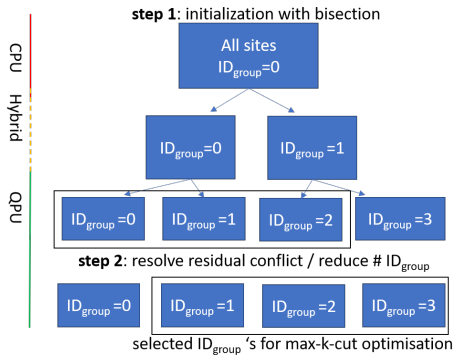


Fig. 2: Schematic of ID_{group} assignment in MCMP algorithm.

$n = 64$ so that each subproblem could be fit (or embedded) in a machine like D-Wave 2000Q. Experiments on different partition strategies did not highlight relevant improvements. By using 6 qubits for the ID_{Cell} assignment, 10 sites can be processed at a time. However, the number of sites with the same ID_{Group} is limited.

C. Monte Carlo Multi-Phases

A more sophisticated and complex algorithm, named Monte Carlo Multi-Phases (MCMP), starts by assigning the ID_{group}, while the ID_{cell} is assigned at later phase (multi-phases). It applies an iterative solution refinement considering randomised subproblems (Monte Carlo) of smaller and tunable size. Each subproblem is formulated as standard max-cut, max-k-cut, or modified max-k-cut as described in the previous section. Another major difference is that the ID_{group} assignment is done exclusively at 'site level' using the particular property of the $\mathbf{S}$ cost matrix: $S_{i,j} = S_{i',j'}$ with $i, i' \in$ same site ν and $j, j' \in$ same site ν'. This allows to consider the cost of having two sites instead of two cells with the same ID_{cell} and, for constant number of QUBO variables, to increase the number of sites per subproblem (roughly by a factor of 3) helping the search of a more global solution for the ID_{cell} assignment. The $\mathbf{S}$ cost matrix can be reduced to a smaller $\mathbf{S}^{site}$ without loss of generality. The drawback is that we need to define, arbitrarily and with some loss of information, a $\mathbf{C}$ cost matrix at a site level, the $\mathbf{C}^{site}$. We make the following choice defining the matrix element $C^{site}_{i,j} = 0$ if the two-sites ID_{cell} assignment subproblem (defined by the submatrix $C_{k,l}$ with k cell id of the i site and l cell id of the j site) can be successfully solved with zero residual cost and $C^{site}_{i,j} = max(C_{k,l})$ otherwise. Since we have only up to 6 nodes and at most a 3 by 3 adjacency matrix, there are only 9 different basic cases to be considered. We have solved exactly and tabulated the results for all cases, thus allowing for a quick construction of $\mathbf{C}^{site}$ without having to solve numerically each subproblem.

1) Phase 1, ID_{group} assignment: the cost function to be minimised in this phase can be written in a compact form as:

$$Tot^{ID_{group}}_{cost} = trace\left(\mathbf{X}^{\top} \cdot \left(\alpha \mathbf{C}^{site} + \beta \mathbf{S}^{site}\right) \cdot \mathbf{X}\right) \qquad (7)$$

where $\mathbf{X}$ is the binary solution matrix of dimension $[\#sites; \#ID_{groups}]$, α and β are free weighting parameters which can be used to prioritise either the costs of PCI or ID_{group} conflict. The ID_{group} assignment phase is further divided in two steps (see figure 2). First, an initialisation assigns ID_{group} by successive bisection (or max-cut) problems starting from the entire set of sites. At the beginning, the number of QUBO variables is large, thus only suitable for classical or hybrid solvers. When the site groups become smaller, a pure QPU solution can be envisaged. The process is terminated either when there are no more residual costs or when all available ID_{group} have been assigned. At this point, the second step refines the solution with an iterative Monte Carlo process; we randomly select a finite set k of ID_{group} values. To each ID_{group} corresponds a group of sites (or simply referred as a 'group') with unresolved conflicts. We then attempt to improve part of the solution by solving a max-k-cut problem with all selected sites as nodes. If the new solution has a lower number of residual conflicts, then it becomes the current best solution. The process continues until all conflicts are solved or an iteration limit is hit or the convergence criteria are satisfied. Finally, and if a minimisation of the ID_{group} usage is requested, the previous process is applied with a minor modification: each max-k-cut problem is solved with $k-1$ colors so that, if successful, an ID_{group} is freed. Again, the process terminates when the predetermined criteria are met. The site ID_{group} is transferred to the cells belonging to a certain site completing this phase and the minimisation of the $\mathbf{S}$ costs.

2) Phase 2, ID_{Cell} assignment: We now consider as nodes in the equivalent graph formulation all cells sharing the same ID_{group}. For each current ID_{group} we build a max-3-cut problem to minimise the related PCI conflict costs in $\mathbf{C}$ with $ID_{Cell} = 1, 2, 3$ as 'colors' of the nodes. If the assignment produces zero residual cost, the scheduling optimisation ends here. However, even if after phase 1 the residual cost $\mathbf{S}^{site}$ was zero, some conflicts may remain and a third final step is attempted.

3) Phase 3, solve residual PCI conflicts: We select the cells belonging to the, normally few, ID_{group} values with residual PCI conflicts and we cluster them into smaller subgroups. For each subgroup of typically few hundreds nodes we assign simultaneously ID_{group} *and* ID_{Cell} using the Single Phase model (see section IV-A). The process is repeated with an increased number of ID_{group} until a zero $\mathbf{C}$ cost solution is found or no more ID_{group} values are available.

D. TIMqual

TIMqual is a Fast Greedy algorithm such as [13]. This algorithm is proprietary of TIM S.p.A., and is the one actually used in the framework of TIM's Open SON framework.

V. Comparison of the Algorithms

For the comparison of the QUBO algorithm we have used a 8-core, 32GB RAM classical computer, using the `qbsolv` tabu QUBO solver included in the `Ocean` Python suite. All

codes are single threaded. The focus of the present work is on the solution quality rather than the computation time, and the ability to reuse the PCIs. From the standpoint of calculation time to elaborate a PCI plan, it is not easy to make a "homogeneous" comparison since the stop criteria of the three algorithms are profoundly different. Thus, all codes are assumed to be run until a suitable convergence criteria is satisfied.

A. ID_{Group} minimisation for medium size PCI networks

We seek a PCI planning for medium size networks with a minimum number of ID_{Group} (PCIs reuse) that nullifies the residual cost associated with the **C** matrix disregarding eventual **S** type conflicts.

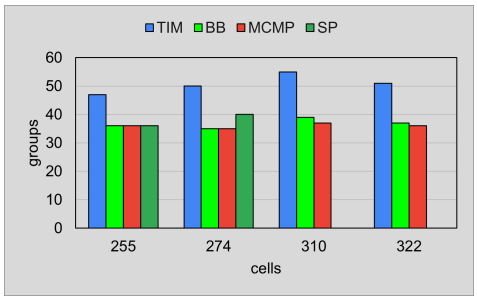


Fig. 3: Comparison chart of the approaches detailed in section III applied to several network sets (up to 322 cells).

The application of QUBO algorithms in the context of PCI planning has provided encouraging results, as can be inferred from the comparison in Fig.3. This chart describes the minimum number of ID_{group} that the four algorithms where able to find. On the X-axis the bars are grouped by the size of the cell set they refer to. All QUBO formulations perform here significantly better than the classical algorithm in *TIMqual* being able to find a good solution with a lower ID_{group}. The SP algorithm, however, can only find a high quality solution for the smallest case, whereas for larger networks it struggles to find a valid solution. This is due to the very large QUBO problem that needs to be solved with tens of thousands variables. It is clear that QUBO algorithms with problem decomposition, such as MCMP and BB, have a good prospect for PCI scheduling of very large networks and only these latter will be considered in the following.

B. Total cost minimisation for large PCI networks

We now turn to the PCI scheduling of macro regions, with urban and non-urban areas. These macro regions have thousands of cells, but the relative percentage of conflicts is lower because of the larger average distance between cells.

As a result, the sparsity of **C** cost matrix is above 90% whilst for **S** is around 80%. Locally, however, there are still dense regions which correspond to cities such Milan and other densely populated areas. The optimisation goal is to nullify the residual PCI conflicts (**C** residual cost) with a minimum number of groups *and* lowest possible ID_{group} type of conflicts (**S** residual cost) which have a negative impact

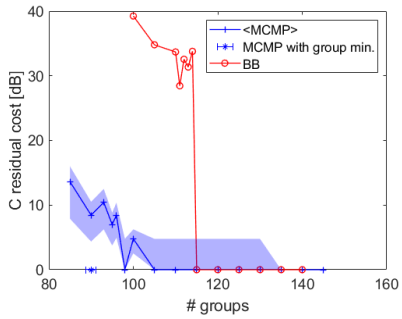


Fig. 4: Macro area with 5718: **C** residual cost in dB as function of the available groups.

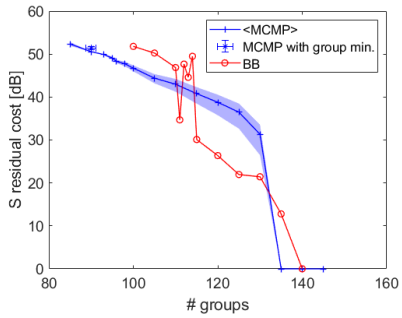


Fig. 5: Macro area with 5718: **S** residual cost in dB as function of the available groups.

on the network as outlined in section III-A. To explore the algorithm behaviour we have scanned the available ID_{group} and looked at the residual costs. The results are plotted in Figures 4 and 5 for the case of 5718 cells networks. The BB algorithm requires substantial CPU time so that no statistics was performed. For the MCMP case, a band representing the variance is displayed (light blue) whereas the blue crosses represent the median value of 10 otherwise identical runs. Both QUBO algorithms can nullify *all* costs using fewer groups than the available ones (168 for 4G). As expected, the general trend is that the lower the available groups, the higher are the residual cost. With group minimisation turned on, the MCMP can find a solution with zero **C** residual cost and only 90 groups but with relatively high **S** residual cost. With group minimisation off, a few PCI conflicts may be left unresolved since the third and last optimisation phase (see section IV-C) is ineffective in this case (no free ID_{group} available). The BB code finds a good scheduling with more groups, although, in some cases, with a lower **C** residual cost. The very small subproblem solved in the BB code probably does not allow for an efficient exploration of the very large configuration space of this network.

These findings have to be compared with the *TIMqual* solution that nullifies the PCI conflicts with a minimum of 119 groups and **C** cost of 26.8dB. For this large and sparse network, it is quite evident the ability of the QUBO-based planners in using fewer groups, especially for the MCMP case.

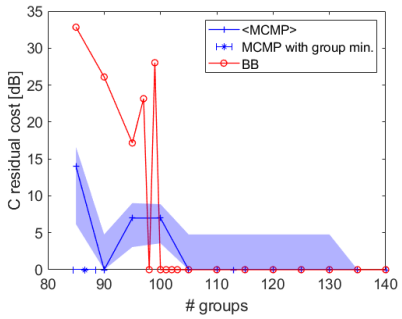


Fig. 6: Macro area with 3085 cells and 10% border cells: **C** residual cost in dB as function of the available groups.

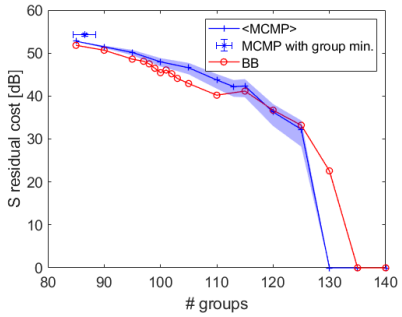


Fig. 7: Macro area with 3085 cells and 10% border cells: **S** residual cost in dB as function of the available groups.

This feature is very promising in view of extremely large areas or for 5G, where many more cells are foreseen, without even yet considering the leverage of quantum acceleration.

1) Managing border cells: When planning PCIs for finite areas, one has often to consider additional constraints, such as the cells at the borders with adjacent areas. In these 'border cells' the PCIs are predefined and fixed to allow a smooth handover of the user. We have implemented the possibility of PCI planning with border cells in both the BB and MCMP code and tested them on a network with a total of 3085 cells and 10% of border cells.

The outcome is shown in figures 6 and 7. The algorithms behave more similarly in this smaller example and both need less groups for a satisfactory solution. More interestingly, they both need a few more groups to nullify the PCI conflicts with respect to the case with no boundary cells (not shown). This result may be understood considering the effect of the boundary cells: on one side they reduce the number of variables but, on the other, introduce many more constraints, de facto reducing the possibility of PCI reuse.

VI. Conclusions

This work has proposed heuristic algorithms formulated as Quadratic Unconstrained Binary Optimization (QUBO) model applied to the PCI assignment problem in the TelCo domain. QUBO-based PCI solver implementations have been proven to often provide better solution quality than classical state-of-the-art solvers and to maximise PCIs reuse.

We have shown, however, that a problem decomposition (either recursive or layered or both) is needed not only for NISQ devices (which are, to date, inherently limited in qubits and connectivity) but also for classical computers when dealing with realistic use cases. Since the decomposition into very small subproblems results in an inefficient resolution for large networks, a possible suggestion could be to use hybrid solvers/decomposers that autonomously determine the best size of problems that can be mapped to a QPU.

Quantum speed-up is hoped and expected when running on the latest D-Wave Pegasus architecture and will be subject of future tests exploiting the proposed problem decomposition. Future work will also include the application of the described approaches to the case of 5G networks.

References

[1] S. Jimaa, K. K. Chai, Y. Chen, and Y. Alfadhl, "Lte-a an overview and future research areas," in *Proceedings of the 2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications*, WIMOB '11, (USA), p. 395–399, IEEE Computer Society, 2011.

[2] H. Yang, A. Huang, R. Gao, T. Chang, and L. Xie, "Interference self-coordination: A proposal to enhance reliability of system-level information in ofdm-based mobile networks via pci planning," *IEEE Transactions on Wireless Communications*, vol. 13, no. 4, pp. 1874–1887, 2014.

[3] R. Acedo-Hernández, M. Toril, S. Luna-Ramírez, and C. Ubeda, "A pci planning algorithm for jointly reducing reference signal collisions in lte uplink and downlink," *Computer Networks*, vol. 119, pp. 112–123, 2017.

[4] F. Xhafa and A. Abraham, "Computational models and heuristic methods for grid scheduling problems," *Future generation computer systems*, vol. 26, no. 4, pp. 608–621, 2010.

[5] S. Khalouli, F. Ghedjati, and A. Hamzaoui, "A meta-heuristic approach to solve a jit scheduling problem in hybrid flow shop," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 5, pp. 765–771, 2010.

[6] F. A. T. Montané and R. D. Galvao, "A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service," *Computers & Operations Research*, vol. 33, no. 3, pp. 595–619, 2006.

[7] H. Kavlak and H. Ilk, "Pci planning strategies for long term evolution networks," in *NETWORKING 2012 Workshops* (Z. Becvar, R. Bestak, and L. Kencl, eds.), (Berlin, Heidelberg), pp. 151–156, Springer Berlin Heidelberg, 2012.

[8] J. Gui, Z. Jiang, and S. Gao, "PCI Planning Based on Binary Quadratic Programming in LTE/LTE-A Networks," *IEEE Access*, vol. 7, pp. 203–214, 2019.

[9] T. Kadowaki and H. Nishimori, "Quantum Annealing in the Transverse Ising Model," *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, vol. 58, pp. 5355–5363, apr 1998.

[10] F. Glover, G. Kochenberger, and Y. Du, "A Tutorial on Formulating and Using QUBO Models," nov 2018.

[11] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, pp. 1–14, feb 2014.

[12] S. Nyberg, "Physical Cell ID Allocation in Cellular Networks," tech. rep., 2016.

[13] B. Charlier and B. Charlier, "The Greedy Algorithms Class: Formalization, Synthesis and Generalization," 1995.