A network-based integer program for physical cell identifier assignment in 4G

Jamie Fairbrother[1] Adam Letchford[2],
in collaboration with Keith Briggs

[1]STOR-i Centre for Doctoral Training, Lancaster University

[2]Department of Management Science, Lancaster University

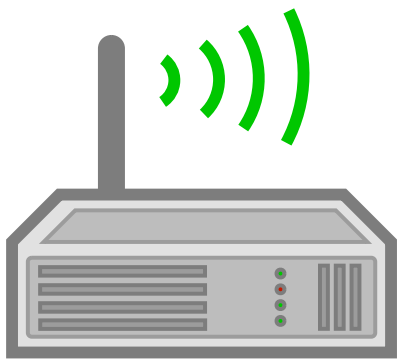
MoN15, Bath, 23rd September 2016

Introduction

Femtocells

Femtocells:

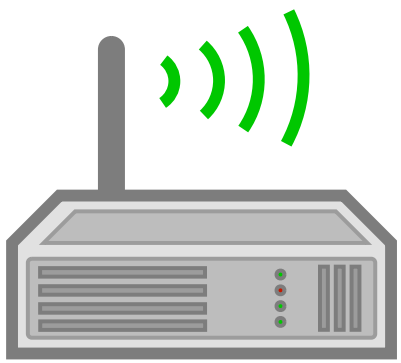
- Femtocells are small home radio devices which use the LTE (4G) network



Femtocells

Femtocells:

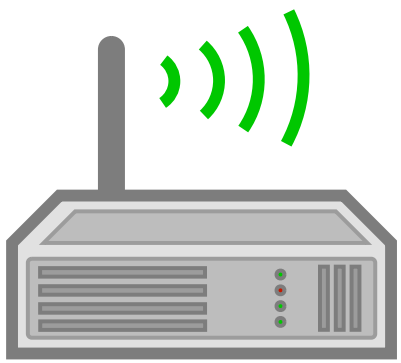
- Femtocells are small home radio devices which use the LTE (4G) network
- Supplement for home WiFi networks



Femtocells

Femtocells:

- Femtocells are small home radio devices which use the LTE (4G) network
- Supplement for home WiFi networks
- Serve BT roaming users to provide faster and more reliable wireless coverage



Physical cell ID (PCI) assignment

- Each cell is assigned a Physical Cell ID (PCI) between 1 and 504

Problem:

Physical cell ID (PCI) assignment

- Each cell is assigned a Physical Cell ID (PCI) between 1 and 504
- Neighbouring cells *must not* have the same PCI

Problem:

Physical cell ID (PCI) assignment

- Each cell is assigned a Physical Cell ID (PCI) between 1 and 504
- Neighbouring cells *must not* have the same PCI
- It is *desirable* for neighbouring devices to have PCIs different up to modulo 3

Problem:

Physical cell ID (PCI) assignment

- Each cell is assigned a Physical Cell ID (PCI) between 1 and 504
- Neighbouring cells *must not* have the same PCI
- It is *desirable* for neighbouring devices to have PCIs different up to modulo 3

Problem:

- Cells are currently assigned a PCI using a distributed dynamic heuristic

Physical cell ID (PCI) assignment

- Each cell is assigned a Physical Cell ID (PCI) between 1 and 504
- Neighbouring cells *must not* have the same PCI
- It is *desirable* for neighbouring devices to have PCIs different up to modulo 3

Problem:

- Cells are currently assigned a PCI using a distributed dynamic heuristic
- It is not known how well this performs in practice

Physical cell ID (PCI) assignment

- Each cell is assigned a Physical Cell ID (PCI) between 1 and 504
- Neighbouring cells *must not* have the same PCI
- It is *desirable* for neighbouring devices to have PCIs different up to modulo 3

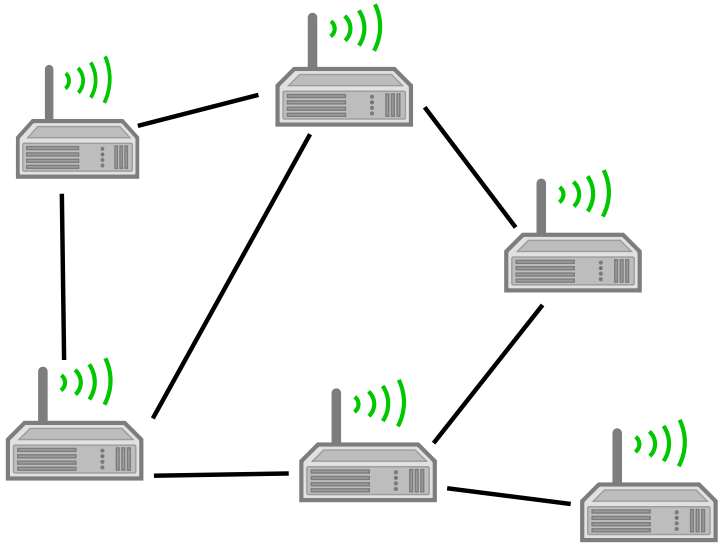
Problem:

- Cells are currently assigned a PCI using a distributed dynamic heuristic
- It is not known how well this performs in practice
- We wish to assess performance via mathematical optimization

Formulation

Modelling the problem

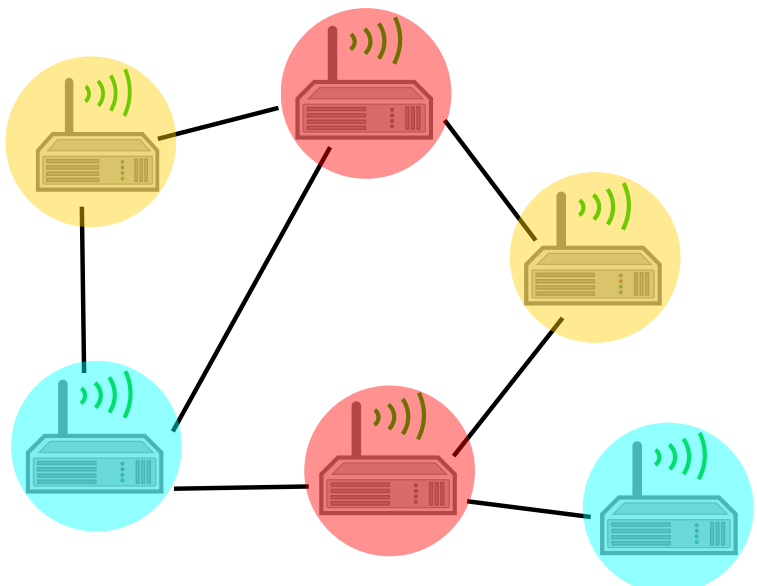
- The system is modelled as a (possibly planar) *sparse* graph $G = (V, E)$ where the nodes correspond to femtocells and an edge is present between two nodes if they are neighbours



Modelling the problem

- The system is modelled as a (possibly planar) *sparse* graph $G = (V, E)$ where the nodes correspond to femtocells and an edge is present between two nodes if they are neighbours

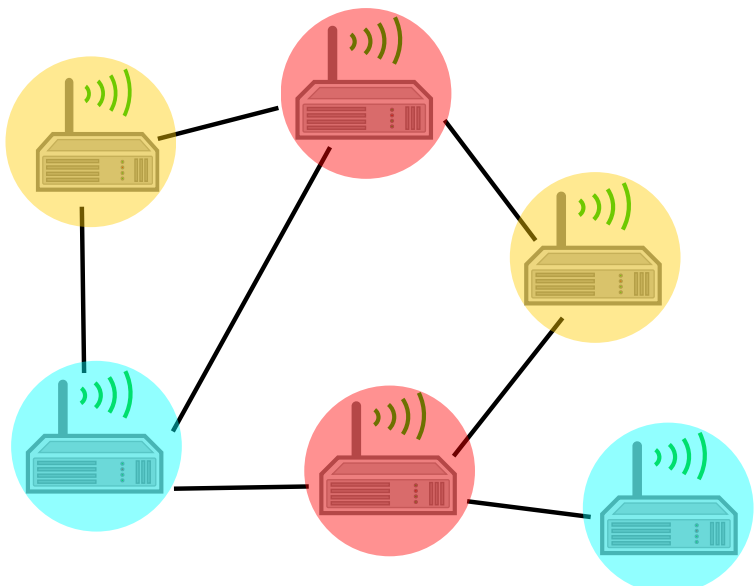
- PCI assignment can be thought of as graph colouring



Modelling the problem

- The system is modelled as a (possibly planar) *sparse* graph $G = (V, E)$ where the nodes correspond to femtocells and an edge is present between two nodes if they are neighbours

- PCI assignment can be thought of as graph colouring
- Each PCI (up to modulo 3) represents a colour and one must assign each node a colour in such a way which avoids neighbouring femtocells using the same colour



Minimizing clashes

- We would like to find a 3-colouring which minimizes the edges whose incident nodes use the same colour
- Can this be done by enumeration?
- No! There are $3^{|V|}$ possible 3-colourings

Integer programming formulation

$$
\begin{aligned}
\min \quad & \sum_{e \in E} y_e & \\
\text{s.t.} \quad & \sum_{c=1}^{k} x_{vc} = 1 & (v \in V) \\
& y_{uv} \geq x_{uc} + x_{vc} - 1 & (\{u, v\} \in E,\ c = 1, \ldots, k) \\
& x_{vc} \in \{0, 1\} & (v \in V,\ c = 1, \ldots, k) \\
& y_{uv} \in \{0, 1\} & (\{u, v\} \in E).
\end{aligned}
$$

where

$$
x_{vc} = \begin{cases} 1 & \text{if node } v \text{ uses colour } c \\ 0 & \text{otherwise} \end{cases}
$$

$$
y_{uv} = \begin{cases} 1 & \text{if nodes } u, v \text{ use the same colour} \\ 0 & \text{otherwise} \end{cases}
$$

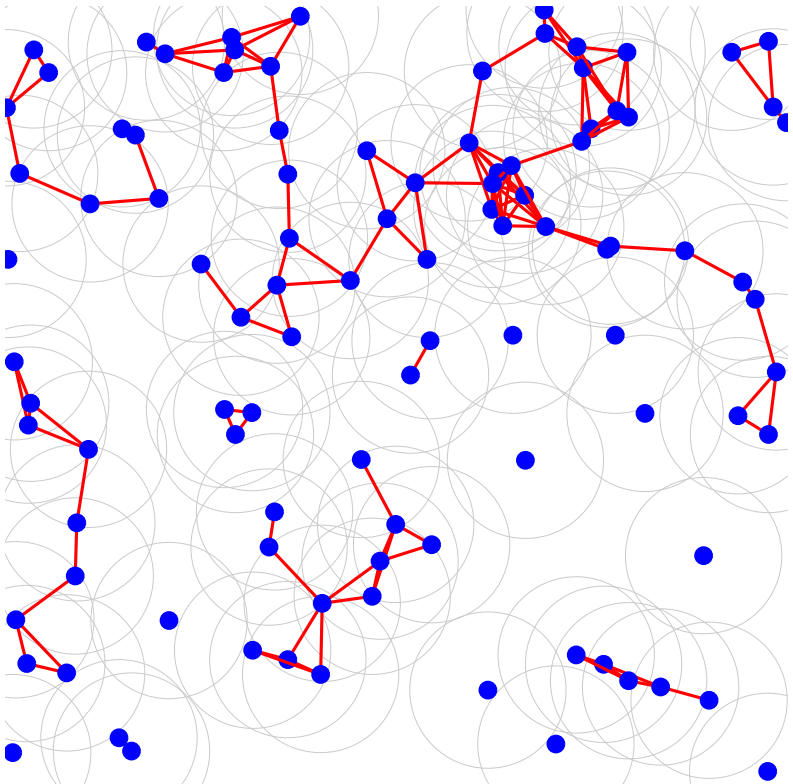
Complexity and solution

- The integer programming formulation given above is a special case of the k-partition problem (k-PP)
- The k-PP is well known to be $\mathcal{NP}$-hard in the strong sense
- We use the following approach:
  1. Preprocessing (reducing and decomposing graph)
  2. Cutting-plane and branch-and-bound

Neighbourhood graph

Random disk graph

- Two radio devices can hear each other if they are within a given radius
- We construct random graphs by sampling points on unit square/torus and linking points within a given radius
- The larger the radius the more dense the graph



Neighbours of neigbour graph

- Conflict also occurs if two femtocells with a common neighbour use the same PCI modulo 3
- Edge set augmented by edges between nodes which have a neighbour in common

Preprocessing

Preprocessing

- Given the difficulty of problem it is essential the graph is small as possible
- The neighbourhood graph can be reduced by two operations: k-core reduction and biconnected component decomposition

k-core reduction

- The k-core of a graph is the largest induced subgraph where all vertices have at least k neighbours:

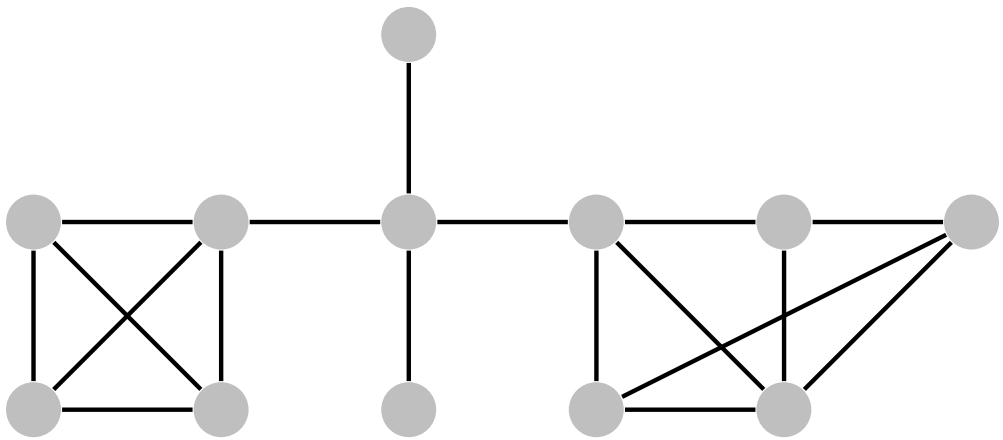


Figure : Full graph

- The k-core yields the same optimal solution value to the k-PP as the original graph

k-core reduction

- The k-core of a graph is the largest induced subgraph where all vertices have at least k neighbours:

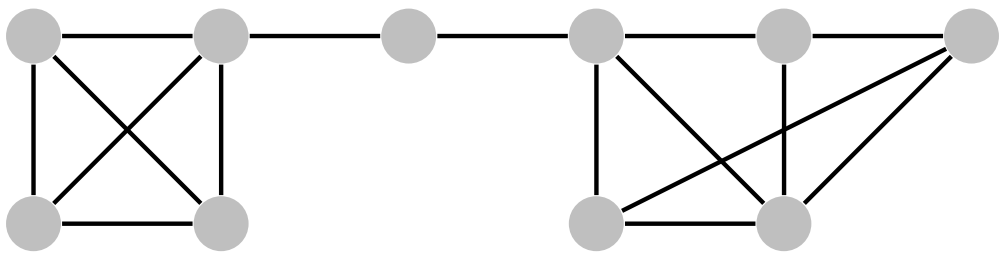


Figure : 2-core reduction

- The k-core yields the same optimal solution value to the k-PP as the original graph

k-core reduction

- The k-core of a graph is the largest induced subgraph where all vertices have at least k neighbours:

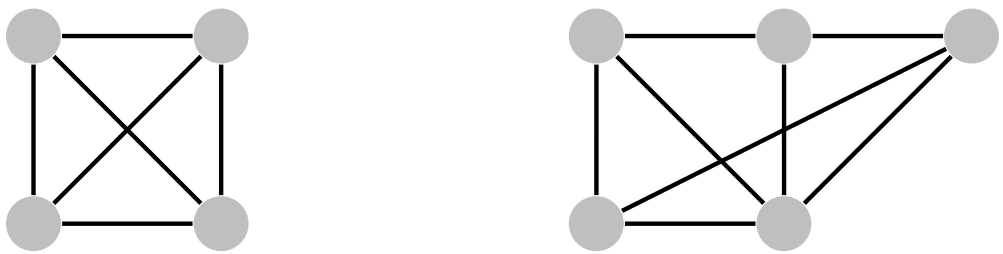


Figure : 3-core reduction

- The k-core yields the same optimal solution value to the k-PP as the original graph

Biconnected components

- A biconnected component is a maximal subgraph which cannot be disconnected by the removal of a single node

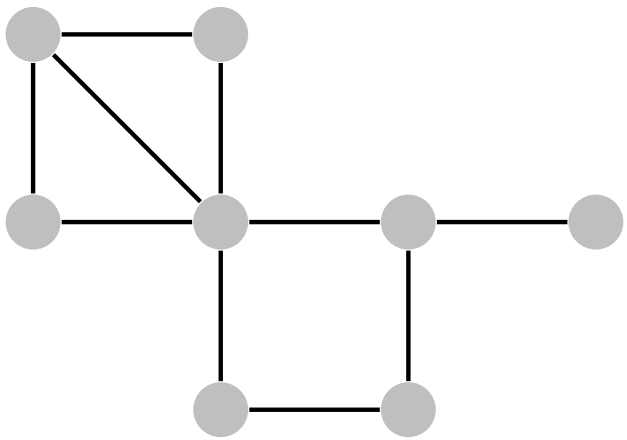


Figure : Full graph

- The optimal solution value to the k-PP is the sum of those for all the biconnected components

Biconnected components

- A biconnected component is a maximal subgraph which cannot be disconnected by the removal of a single node

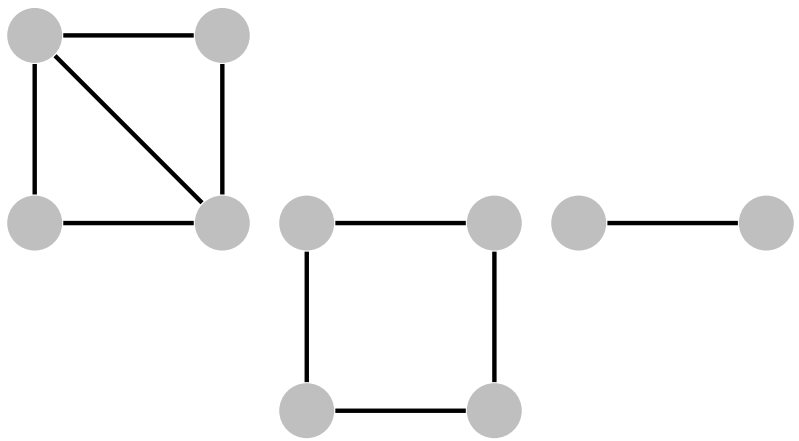


Figure : Biconnected components

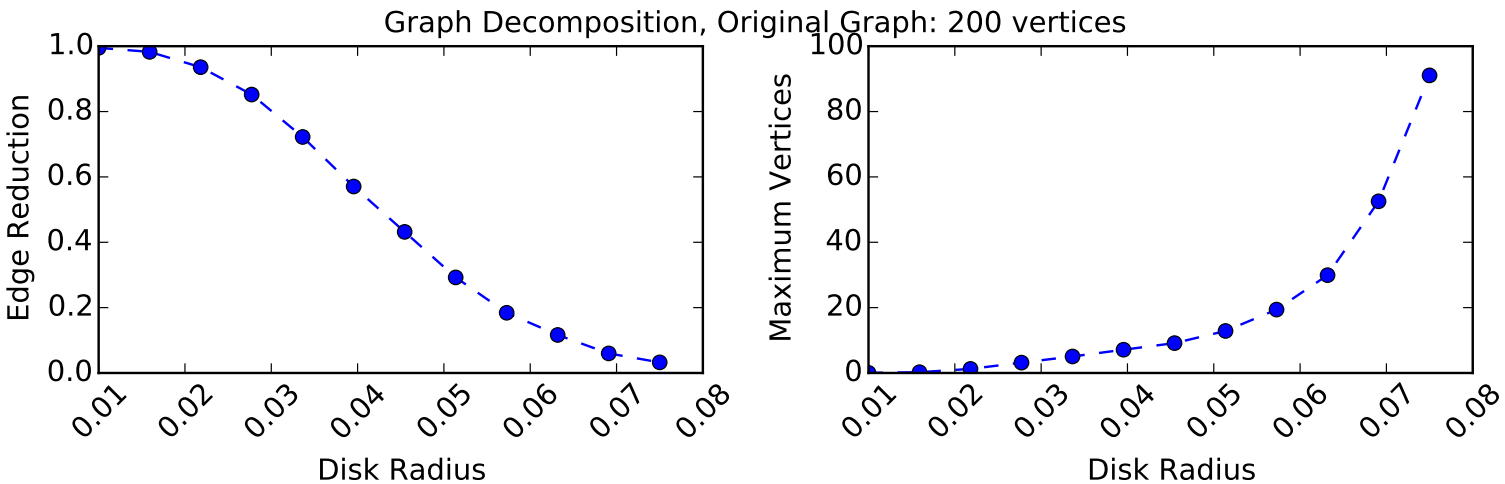
- The optimal solution value to the k-PP is the sum of those for all the biconnected components

Graph decomposition for PCI assignment

- k-core reduction can result in graph which is not connected or biconnected, and biconnected components may not be reducible
- 3-core reductions and biconnected components decompositions can be applied iteratively
- The optimal solution value for k-PP is the sum of that for all components

Power of decomposition

- The reduction achieved depends the sparsity of the graph



Cutting-Plane Algorithm

Recap: Integer programs and linear relaxations

$$\begin{aligned} \underset{x}{\text{minimize}}\ & c^T x \\ \text{s.t. } & Ax \geq b \\ & x \in \mathbb{Z}^n \end{aligned}$$

- The *linear relaxation* of an integer program is the problem without integer constraints
- A *cutting-plane* is an inequality which is satisfied by all feasible integer solutions but violated by at least one solution of the linear relaxation

Recap: Integer programs and linear relaxations

$$\begin{aligned} \underset{x}{\text{minimize}}\ & c^T x \\ \text{s.t. } & Ax \geq b \\ & x \in \mathbb{R}^n \end{aligned}$$

- The *linear relaxation* of an integer program is the problem without integer constraints
- A *cutting-plane* is an inequality which is satisfied by all feasible integer solutions but violated by at least one solution of the linear relaxation

Recap: Cutting-planes algorithm

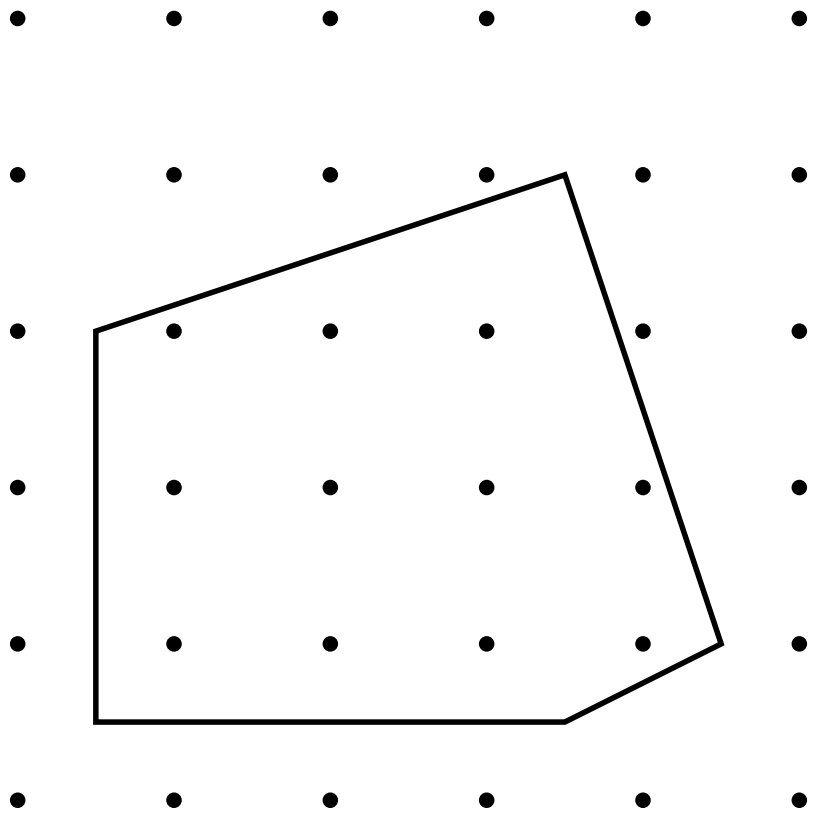


Figure : Cutting-plane algorithm

Recap: Cutting-planes algorithm

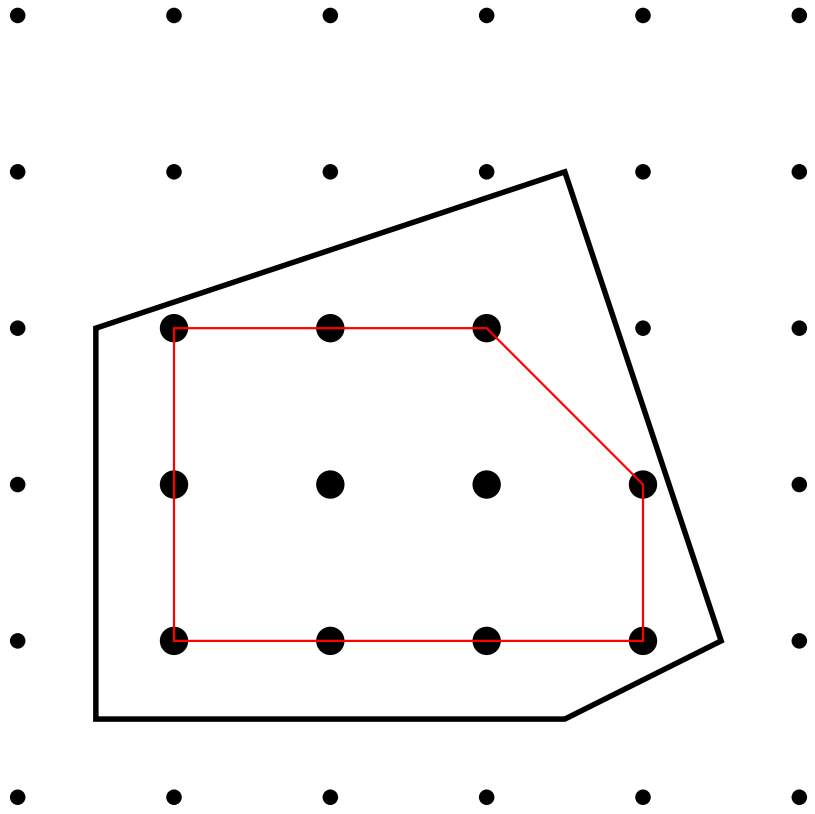


Figure : Cutting-plane algorithm

Recap: Cutting-planes algorithm

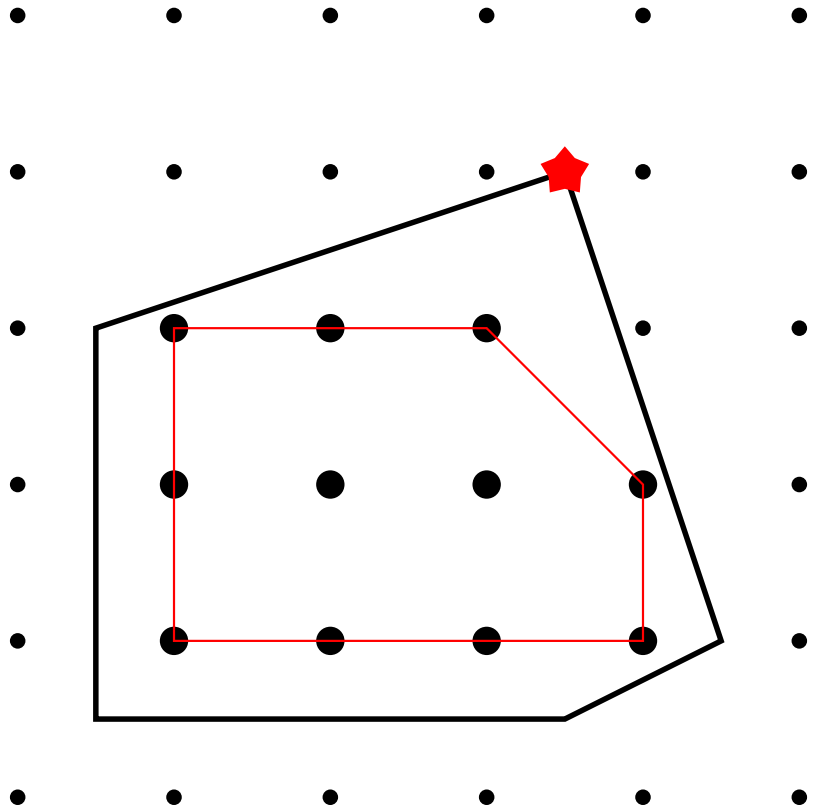


Figure : Cutting-plane algorithm

Recap: Cutting-planes algorithm

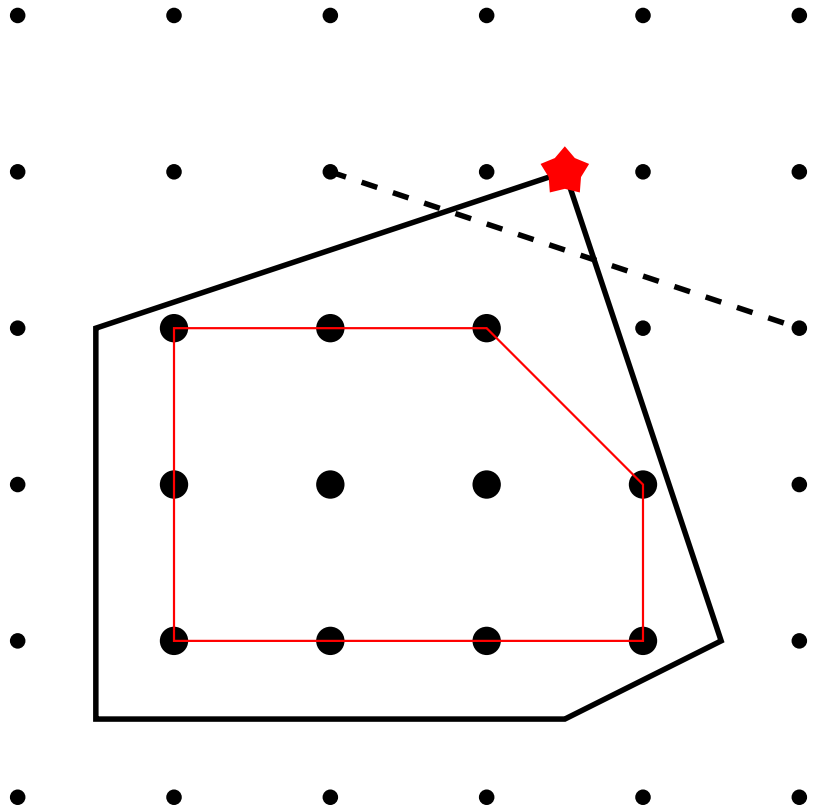


Figure : Cutting-plane algorithm

Recap: Cutting-planes algorithm

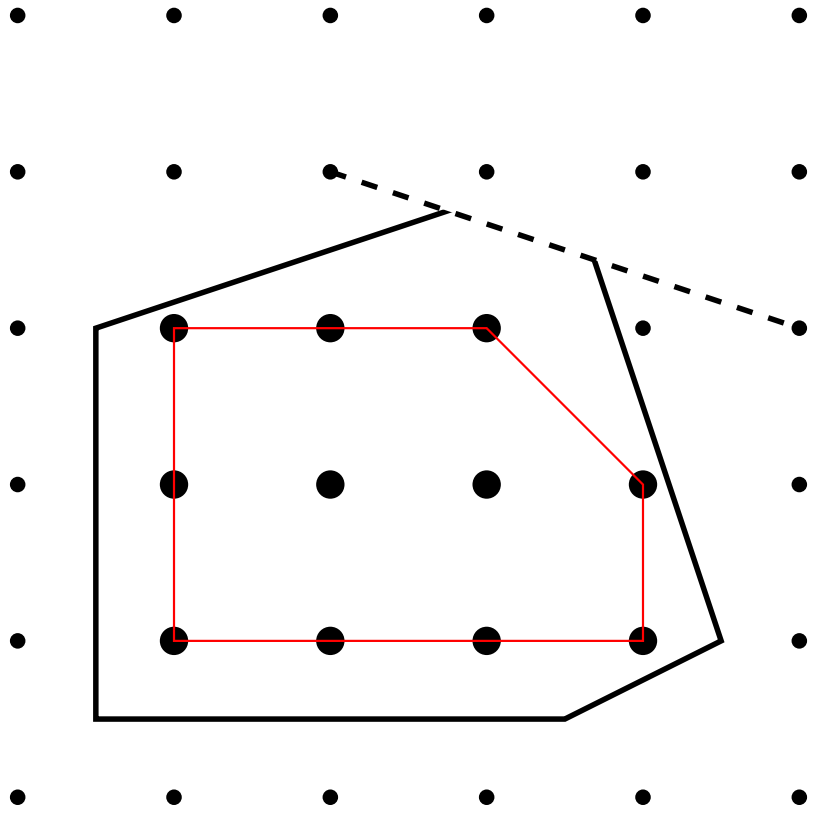


Figure : Cutting-plane algorithm

Recap: Cutting-planes algorithm

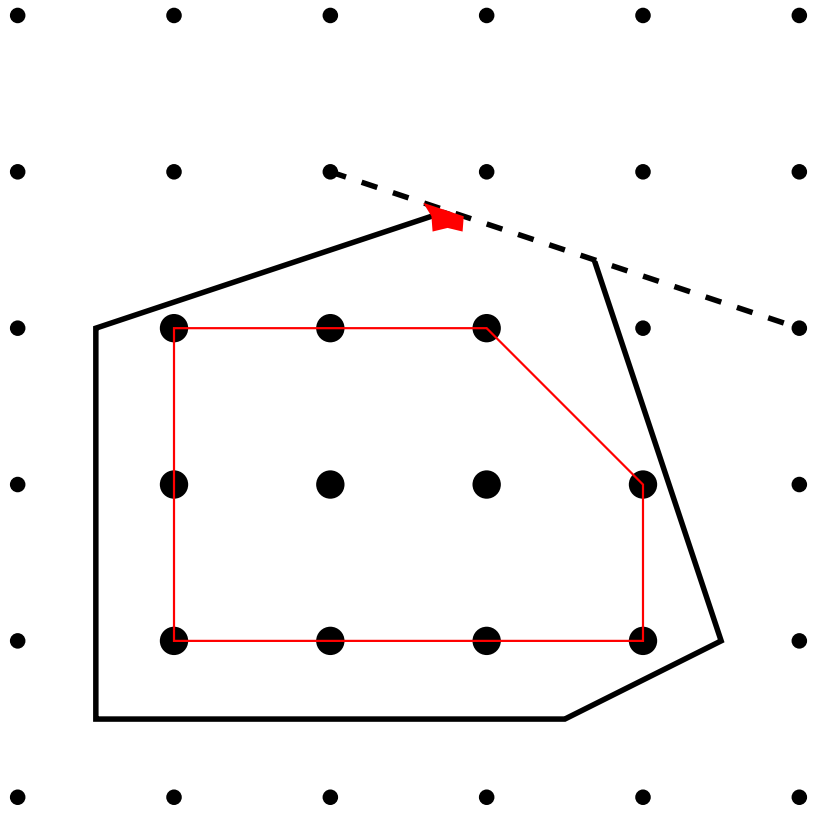


Figure : Cutting-plane algorithm

Recap: Cutting-planes algorithm

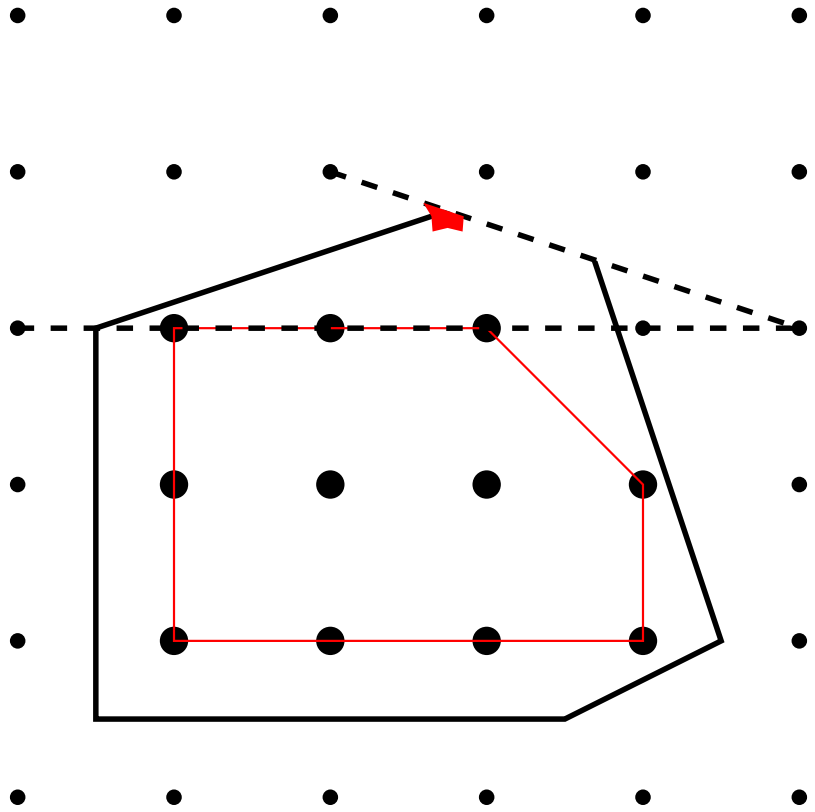


Figure : Cutting-plane algorithm

Recap: Cutting-planes algorithm

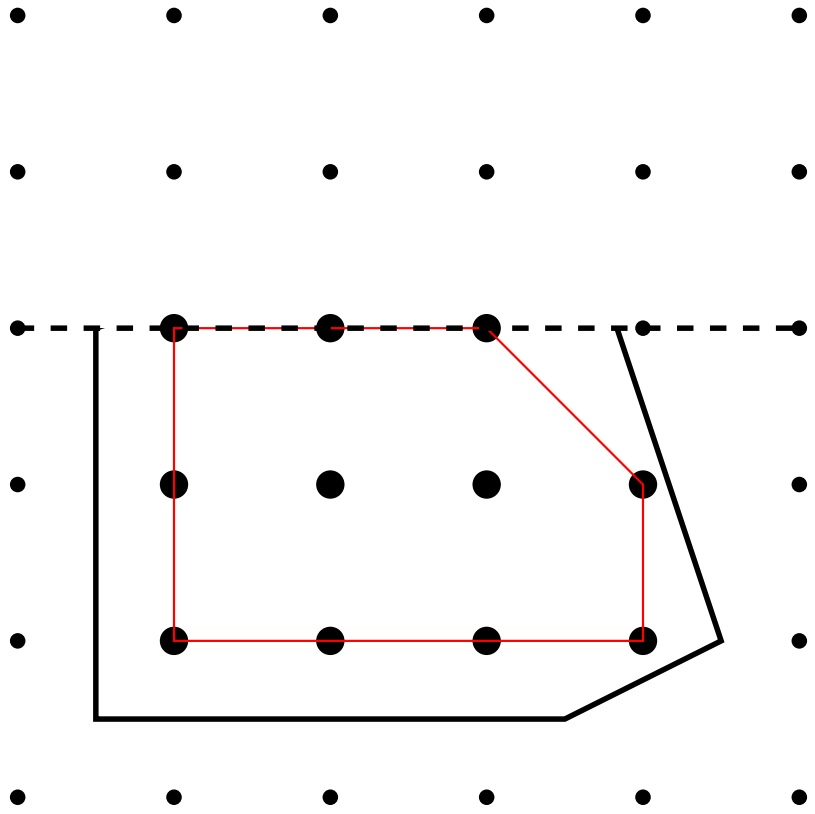


Figure : Cutting-plane algorithm

Recap: Cutting-planes algorithm

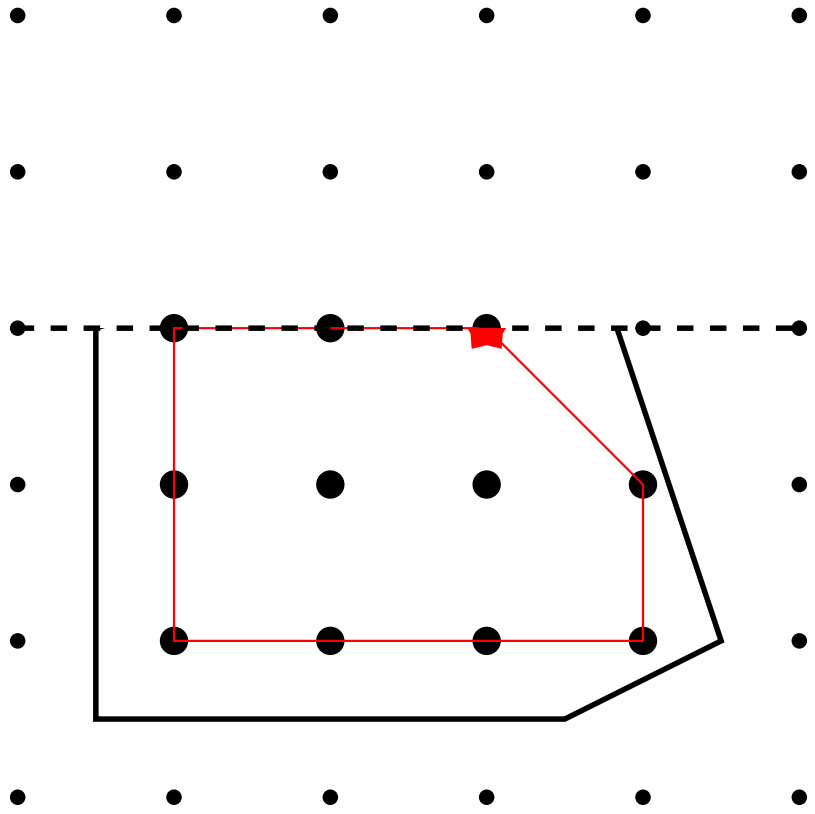


Figure : Cutting-plane algorithm

Clique inequalities

Theorem ([Chopra and Rao, 1993])

For a clique $C \subset V$, the following inequality is valid:

$$\sum_{u,v \in C} y_{uv} \geq \binom{t+1}{2} r + \binom{t}{2} (k - r)$$

where $t = \lfloor \frac{|C|}{k} \rfloor$ and $r \bmod k$. Moreover, it is facet-defining if $r \neq 0$.

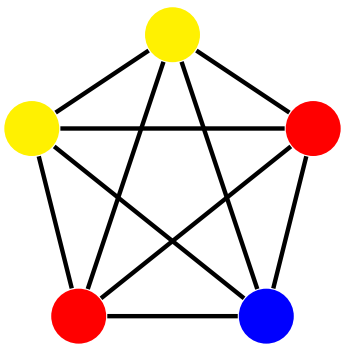


Figure : A clique of size 5

Solution Algorithm

1. Run an LP-based cutting-plane algorithm in y-space with clique inequalities
2. Delete non-binding cuts
3. Add the x variables and associated constraints
4. Run branch-and-bound on the strengthened (x, y) formulation

Modulo-6 Extension

Modulo-6 Interference

- Adjacent devices which both use the same PCI modulo 6 can cause additional interference
- Taking this into account our problem becomes a 6-colouring

Problem reformulation

$$
\begin{aligned}
\min \quad & w \sum_{\{u,v\}\in E} y_{uv} + \sum_{\{u,v\}\in E} z_{uv} & \\
\text{s.t.} \quad & \sum_{c=1}^{6} x_{vc} = 1 & (v \in V) \\
& y_{uv} \geq x_{uc} + x_{u,c+3} + x_{vc} + x_{v,c+3} - 1 & (\{u,v\} \in E,\ c = 1,2,3) \\
& z_{uv} \geq x_{uc} + x_{vc} - 1 & (\{u,v\} \in E,\ c = 1,\ldots,6) \\
& x_{vc} \in \{0,1\} & (v \in V,\ c = 1,\ldots,6) \\
& y_{uv} \in \{0,1\} & (\{u,v\} \in E) \\
& z_{uv} \in \{0,1\} & (\{u,v\} \in E).
\end{aligned}
$$

A valid inequality

Theorem ([Fairbrother and Letchford, 2016])

For all $C \subseteq V$ inducing a clique in G, with $|C| \geq 3$, the following "(y, z)-clique" inequalities are valid:

$$2 \sum_{u,v \in C} z_{uv} \geq \sum_{u,v \in C} y_{uv} - \left\lfloor \frac{|C|}{2} \right\rfloor .$$

A new cutting-plane algorithm

1. Run an LP-based cutting-plane algorithm in y-space.
2. Delete non-binding cuts.
3. Add the z variables and associated constraints.
4. Run an LP-based cutting-plane algorithm in (y, z)-space.
5. Delete non-binding cuts.
6. Add the x variables and associated constraints.
7. Run branch-and-bound on the strengthened (x, y, z) formulation.

Numerical results: Optimality Gaps

- Lower bounds calculated after each cutting-plane phase for random neighbourhood graphs

max. clique size	y-cut opt. val.	yz-cut opt. val.	optimum value
6	17.250000	17.250000	18.0
8	27.000000	30.500000	32.0
7	24.500000	25.500000	26.0
8	33.831933	39.019776	41.0
8	31.767464	35.262257	37.0

Table : Optimality gaps for random disk graphs with 50 points and radius 0.15

Numerical results: Solution time

- Time to solve problem with and without preprocessing and cutting plane are recorded

BB	CP+BB	PP+BB	PP+CP+BB
1.306422	0.341890	2.664614	0.277587
33.182424	2.140248	11.791344	0.713823
166.210727	6.849852	49.542621	9.210316
34.594568	0.985396	22.054915	0.621656
1.455401	0.410856	2.033473	0.388565

Table : Solution times for random disk graphs with 50 points and radius 0.15
(BB=Branch-and-bound, CP=Cutting Plane, PP=Preprocessing)

Conclusions

Conclusions

- The problem of assigning PCI to femtocell devices can be formulated as an integer program
- This problem is defined over an appropriately defined network
- Problem is difficult especially when one takes into account modulo 6 clashes
- Preprocessing is particularly effective at reducing problem for more sparse graphs
- Cutting-plane algorithms based on clique inequalities yield good lower bounds and drastically reduce solution time

Future work

- Test performance of distributed heuristic with respect to optimal solution value
- Use more realistic point processes, or even real data for construction of neighbourhood graph
- Consider other possible extensions, such as power configuration


Chopra, S. and Rao, M. (1993).
The partition problem.
Mathematical Programming, 59(1-3):87–115.


Fairbrother, J. and Letchford, A. (2016).
Some variants of the k-partition problem arising in mobile wireless communications.
Working paper.