Two Papers on Online Load Balancing of Temporary Jobs

Alex Cann

April 11, 2021

Online Load Balancing of Related Machines with Temporary Jobs

Amotz Bar-Noy, Ari Freud, Joseph Naor

April 11, 2021

Related Machines

Definition

- a set of machines $\{m_1 \ldots\}$
- each machine m_i has a speed v_i
- W.L.O.G if $i < j$ then $v_i > v_j$
- Each event consists of a Job arriving or leaving
- Each j consists of a weight w_j
- The load of m_i is the sum of the weights of jobs assigned to that machine divided by v_i.

Competitive Ratio

Definition

- $\mathcal{J}_i$ is the set of jobs from $\mathcal{J}$ active at time i
- **COST**$(\mathcal{J})$ is the maximum load on any machine at any point in the assignment of $\mathcal{J}$
- **COST**$(j) = max_{1 \leq i \leq j}\{$**COST**$(\mathcal{J}_i)\}$

Previous Results

Deterministic

Competitive ratio of 20 using SLOW-FIT algorithm by Azar et al [1]

Randomized

Randomizing SLOW-FIT gives competitive ratio of 13.59 [2]

Informal Heuristics

Definition (Eligibility)

The machine is fast enough for the job.

Definition (Saturation)

The machine is too busy with current jobs.

Formal Heuristics

Define two constants l and s

Definition (Eligibility)

A machine m_i is eligible for a job j if $w_j/v_i \leq l \cdot \mathbf{OPT}(j)$. We say that a job is *permitted* on the set of machines for which it is eligible.

Definition (Saturation)

A machine m_i is saturated if the load at time j exceeds $s \cdot \mathbf{OPT}(j)$.

Algorithm

Algorithm PushRight

Assign each job to the rightmost(slowest) unsaturated eligible machine.

Proof of Competitive Ratio.

Assume that there is always some unsaturated eligible machine when each job j arrives.
Then j is assigned to a machine m such that
$\mathbf{LOAD}(m) < s \cdot \mathbf{OPT}(j)$ and $w_j/v < l \cdot \mathbf{OPT}(j)$.
Thus $\mathbf{COST}(j) \leq (s+l) \cdot \mathbf{OPT}(j) \leq (s+l) \cdot \mathbf{OPT}$ □

Proof of Soundness

Lemma

let $s \geq 4$ *and let* $\{a_i\}_{i=0}^{\infty}$ *be any sequence of numbers such that* $\forall i$

1. $a_0 = 0$
2. $a_1 > 0$
3. $a_{i+2} \geq s(a_{i+1} - a_i)$

Then $\forall i, s(a_{i+1} - a_i) > a_{i+1}$

Proof of Soundness(Flawed)

Theorem (Spurious)

if $s \geq 4$, then whenever a job arrives at-least one of it's eligible machines is unsaturated.

For contradiction assume that some job arrives and all of its eligible machines are saturated.
Construct a sequence of machines $\{m_i\}_{i=0}^{\infty}$, jobs $\{j_i\}_{i=0}^{\infty}$ and speed sums $\{V_i\}_{i=0}^{\infty}$ such that j_0 is the first such job and:

1. $V_i = \sum_{k=1}^{m_i} v_k$
2. $\forall i, V_{i+2} \geq s(V_{i+1} - V_i)$
3. $m_1 > 0$ and $\{m_i\}$ increases monotonically. $m_0 = 0$ for convenience
4. $\forall i$ job j_i is *permitted* on $m_i + 1, \ldots m_{i+1}$ but j_i is not assigned to the right of m_i. ($m_i + 1, \ldots m_{i+1}$ are all saturated)
5. $\forall i, j_{i+1}$ precedes j_i in $\mathcal{J}$

Proof Outline

Property 5 asserts that j_0 is preceded by an infinite number of jobs. (A contradiction)

Property 1 holds by construction

Property 5 holds by construction

Property 4 $\Rightarrow s \cdot \mathbf{OPT}(j_i) \sum_{k=m_i+1}^{m_{i+1}} v_k \leq \mathbf{OPT}(j_i) \sum_{k=1}^{m_{i+2}} v_k \quad *$

$* \Rightarrow 2$

Properties 2, 1, $m_1 > 0$ and Lemma $\Rightarrow V_{i+1} < s(V_{i+1} - V_i) \quad **$

1 and $** \Rightarrow 3$

Property 1 is the definition of $\{V_i\}$ from $\{m_i\}$.
Construct $\{m_i\}_{i=0}^{\infty}$, $\{j_i\}_{i=0}^{\infty}$ inductively
Let m_{i+1} be the rightmost machine eligible for job j_i. Then given m_{i+1} and j_i we can define j_{i+1} and m_{i+2}.
By Property 1 and 3 Lemma 1 applies to V_i and thus $V_{i+1} < s(V_{i+1} - V_i)$. By property 4 since $m_i + 1 \ldots m_{i+1}$ are all saturated they each have weight $\geq s \cdot \mathbf{OPT}(j_i)$.
Then, total weight on $m_i + 1$ to $m_{i+1} \geq s \cdot \mathbf{OPT}(j_i) \sum_{k=m_i+1}^{m_{i+1}} v_k$

let J denote the set of jobs assigned to $\{m_i + 1 \ldots m_{i+1}\}$.
let $\mathcal{A}$ be an optimal assignment of all active jobs at time j_i.
Define m_{i+2} to be the rightmost machine $\mathcal{A}$ assigns some $j \in J$.
Define j_{i+1} to be one such job assigned by $\mathcal{A}$. Note that j_{i+1} precedes j_i (Therefore Property 5 holds).
Since $\mathcal{A}$ is optimal every machine m has weight $\leq v_m \cdot \mathbf{OPT}(j_i)$
Therefore,

$$\begin{aligned} s \cdot \mathbf{OPT}(j_i) \sum_{k=m_i+1}^{m_{i+1}} v_k &\leq \text{total weight on } m_i + 1 \ldots m_{i+1} \\ &\leq \text{total weight on } m_1 \ldots m_{i+2} \\ &\leq \mathbf{OPT}(j_i) \sum_{k=1}^{m_{i+2}} v_k \end{aligned}$$

From property 1 and above equation:

$$\begin{aligned}\sum_{k=1}^{m_{i+1}} v_k =& V_{i+1} \\ &< s(V_{i+1} - V_i) \\ &= s \sum_{k=m_i+1}^{m_{i+1}} v_k \\ &\leq \sum_{k=1}^{m_{i+2}} v_k \\ &= V_{i+2}\end{aligned}$$

Therefore property 3 holds. ($m_{i+2} > m_{i+1}$ and $V_{i+2} \geq s(V_{i+1} - V_i)$)

Property 4 does not hold because we use $\mathbf{OPT}(j_i)$ while determining whether j_{i+1} is permissible for m_{i+2}.
$\mathbf{OPT}(j_{i+1}) \leq \mathbf{OPT}(j_i)$ so m_{i+2} may only become eligible after j_{i+1} was scheduled.

Proof of Soundness(Fixed)

Theorem (Fixed)

If $s - (s + l)/l \geq 4$ then whenever a job arrives at least one of its eligible machines is unsaturated.

Replace property 2 with $\forall i, V_{i+2} \geq 4(V_{i+1} - V_i)$
Define $J' = \{j \in J \mid \mathbf{OPT}(j_i) \leq l\mathbf{OPT}(j)\}$
Define m_{i+2} to be the rightmost machine $\mathcal{A}$ assigns some $j \in J'$.
Define $j_{i+1} \in J'$ to be one such job assigned by $\mathcal{A}$.
Now since $\mathbf{OPT}(j_i) \leq l\mathbf{OPT}(j_{i+1})$ by definition of J', m_{i+2} is eligible for j_{i+1}

as before

$$
\begin{aligned}
4 \cdot \mathbf{OPT}(j_i) \sum_{k=m_i+1}^{m_{i+1}} v_k <& s \cdot \mathbf{OPT}(j_i) \sum_{k=m_i+1}^{m_{i+1}} v_k \\
\leq& \text{total weight on } m_i + 1 \ldots m_{i+1} \\
\leq& \text{total weight on } m_1 \ldots m_{i+2} \\
\leq& \text{total weight of jobs in } J
\end{aligned}
$$

Assume $J' \neq J$ and let j^* be the last arriving job in $J - J'$. By definition of J', $l\mathbf{OPT}(j^*) < \mathbf{OPT}(j_i)$. Therefore, immediately after j^* has been assigned

$$\begin{aligned}\text{The maximum weight on a machine} &= \max_m \mathbf{LOAD}(m) \cdot v_m \\ &\leq (s + l)\mathbf{OPT}(j^*) \\ &< (1/l)(s + l)\mathbf{OPT}(j_i)\end{aligned}$$

$$
\begin{aligned}
(1/l)(s+l)\mathbf{OPT}(j_i)\sum_{k=m_i+1}^{m_{i+1}} v_k &> \text{total weight on } m_i+1 \dots m_{i+1} \\
&\geq (s+l)\mathbf{OPT}(j^*)\sum_{k=m_i+1}^{m_{i+1}} v_k \\
&> 4\mathbf{OPT}(j^*)\sum_{k=m_i+1}^{m_{i+1}} v_k \\
&= 4\mathbf{OPT}(j^*)(V_{i+1}-V_i) \\
&> \mathbf{OPT}(j^*)V_{i+1} \qquad \text{(By lemma 1)} \\
&= \mathbf{OPT}(j^*)\sum_{k=1}^{m_{i+1}} v_k
\end{aligned}
$$

$$(1/l)(s+l)\mathbf{OPT}(j_i)\sum_{k=m_i+1}^{m_{i+1}} v_k > \text{total weight on } m_i+1\ldots m_{i+1}$$

$$> \mathbf{OPT}(j^*)\sum_{k=1}^{m_{i+1}} v_k$$

$$\geq \sum_{j\in J-J'} w_j$$

By definition of **OPT**, $\forall j \in J - J', \mathbf{OPT}(j) \leq \mathbf{OPT}(j^*)$

$$\text{total weight of jobs in } J' = \sum_{j \in J} w_j - \sum_{j' \in J - J'} w_{j'}$$

$$> (s - (s+l)/l)\mathbf{OPT}(j_i) \sum_{k=m_i+1}^{m_{i+1}} v_k$$

$$\geq 4\mathbf{OPT}(j_i) \sum_{k=m_i+1}^{m_{i+1}} v_k$$

Additionally, since $\mathcal{A}$ assigns $j \in J'$ to machines $1 \ldots m_{i+2}$

$$\text{total weight of jobs in } J' < \mathbf{OPT}(j_i) \sum_{k=1}^{m_{i+2}} v_k$$

Therefore,

$$\begin{aligned} 4\mathbf{OPT}(j_i) \sum_{k=m_i+1}^{m_{i+1}} v_k &< (s-(s+l)/l)\mathbf{OPT}(j_i) \sum_{k=m_i+1}^{m_{i+1}} v_k \\ &< \text{total weight of jobs in } J' \\ &< \mathbf{OPT}(j_i) \sum_{k=1}^{m_{i+2}} v_k \end{aligned}$$

From property 1 and above equation:

$$\begin{aligned}
\sum_{k=1}^{m_{i+1}} v_k &= V_{i+1} \\
&< 4(V_{i+1} - V_i) \\
&= 4 \sum_{k=m_i+1}^{m_{i+1}} v_k \\
&\leq \sum_{k=1}^{m_{i+2}} v_k \\
&= V_{i+2}
\end{aligned}$$

Theorem

For $s = 5 + \sqrt{5}$ and $l = 1 + \sqrt{5}$, Algorithm PushRight is sound and guarantees competitive ratio of $s + l = 6 + 2\sqrt{5} \approx 10.47$

Use a common technique called doubling to turn the deterministic algorithm into a randomized one.
Deterministic doubling algorithm provides a weaker bound then PushRight
The bound improves when Randomness is introduced.

Doubling PushRight

Definition (Eligibility)

When job j with weight w arrives machine i is eligible if $w/v_i \leq GUESS$

Definition (Saturation)

A machine i is saturated if the total weight of jobs assigned to it since $GUESS$ was last doubled is $\geq 4\mathbf{OPT}(j)$

Algorithm 1 Doubling PushRight Algorithm

```
procedure DOUBLINGPUSHRIGHT
Initialize: When the first job arrives:
    Initialize GUESS ← w/v_1
To assign job j do:
    while GUESS < OPT(j) do
        GUESS ← k · GUESS
    Assign j to the rightmost unsaturated eligible machine
```

Theorem (Doubling)

Whenever a job arrives at least one of its eligible machines is unsaturated.

Let J from the spurious proof be the set of active jobs between the doubling of *GUESS* and the arrival of j_i.
let g be the cost of *GUESS* during this period.
Optimal assignment $\mathcal{A}$ has cost $\leq \mathbf{OPT}(j_i) \leq g$ and j_{i+1} is placed on m_{i+2}.
Then, $\frac{w}{v_{m_{i+2}}} \leq g$ and m_{i+2} is eligible for job j_{i+1}

Theorem

Algorithm Doubling PushRight is ≈ 14.47 *competitive*

Algorithm 2 Randomized PushRight Algorithm

1: **procedure** RANDOMIZEDPUSHRIGHT
Initialize: When the first job arrives:
2: r $\leftarrow$ RAND$(0,1)$
3: Initialize $GUESS \leftarrow k^{r-1} \cdot w/v_1$
To assign job j do:
4: **while** $GUESS <$ **OPT**(j) **do**
5: $GUESS \leftarrow k \cdot GUESS$
6: Assign j to the rightmost unsaturated eligible machine

Theorem

Algorithm Random Doubling PushRight is ≈ 9.572 *competitive for an oblivious adversary*

Channel Assignment Problem in Cellular Networks

Pilu Crescenzi, Giorgio Gambosi, Paolo Penna

April 11, 2021

Load balancing of Temporary jobs on Restricted Machines

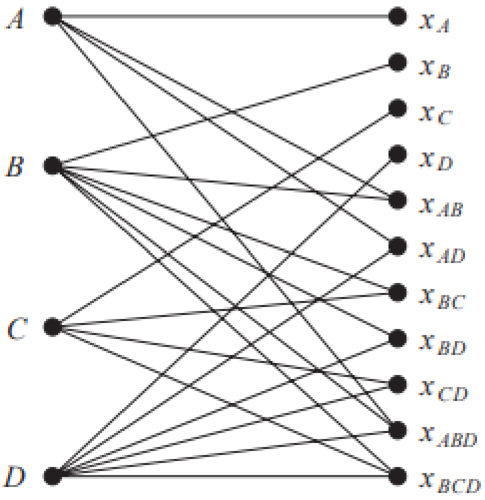
Definition

- a set of machines $\mathcal{M} = \{m_1, \ldots, m_n\}$.
- let $\mathcal{T} \subseteq 2^{\mathcal{M}}$ be a set of job types
- A job type represents the set of processors it can be scheduled on
- Each job j consists of a weight w_j and a job type $x_j \in \mathcal{T}$ the set of machines it can be assigned to.
- The load of m_i is the sum of the weights of jobs assigned to that machine.

Associated Bipartite Graph

Can create an associated bipartite graph for any restricted assignment problem:

$$
\begin{aligned}
G &= (X \cup P, E) \\
P &= \text{The set of Processors} \\
X &= \{x \mid x \text{ is a job type}\} \\
E &= \{(x, y) \mid \text{ job type } x \text{ is assignable to processor } y\}
\end{aligned}
$$

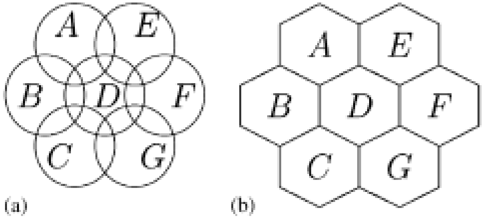

A
B
C
D
x_A
x_B
x_C
x_D
x_{AB}
x_{AD}
x_{BC}
x_{BD}
x_{CD}
x_{ABD}
x_{BCD}

Cellular Network Channel Assignment Problem

Definition

- Given a set of overlapping 2 dimensional circular cells (a.k.a. base stations)
- At any point at most 3 cells overlap
- Receive communication requests r consisting of points p_r and bandwidth b_r
- Each r must be served by one of the cells including p_r
- requests can move themselves from one point to another
- Simulated by the original job ending and a new job arriving whenever a cell border is reached.

Cellular Network Channel Assignment Problem



Reduce Channel Assignment to Restricted Machines

To create a corresponding instance of the restricted assignment problem:

- Each Base Station is a processor
- One job type for the interior of each hexagon
- One job type for each edge between 2 hexagons
- One job type for each vertex between 3 hexagons
- request position $\Rightarrow$ job type
- request bandwidth $\equiv$ job weight

For the channel assignment problem the bipartite graph is:

$$
\begin{aligned}
G =& (X \cup P, E) \\
P =& \text{the set of Base Stations} \\
X =& \{x_A \mid A \in P\} \\
& \cup \{x_{AB} \mid A, B \in P, \quad \text{A,B share an edge}\} \\
& \cup \{x_{AB} \mid A, B, C \in P, \quad \text{A,B,C share a vertex}\} \\
E =& \{(x_A, A) \mid x_A \in X\} \\
& \cup \{(x_{AB}, y) \mid x_{AB} \in X, \quad y \in \{A, B\}\} \\
& \cup \{(x_{ABC}, y) \mid x_{ABC} \in X, \quad y \in \{A, B, C\}\}
\end{aligned}
$$

Intuition

Consider On-line load balancing of identical machines with temporary jobs.
Since every job can be assigned to every machine, we can create the complete associated bipartite graph:

$$\begin{aligned} G =& (X \cup P, E) \\ P =& \text{the set of Processors} \\ X =& \{x \mid x \in \mathcal{T}\} \\ E =& \{(x, y) \mid x \in X, y \in P\} \end{aligned}$$

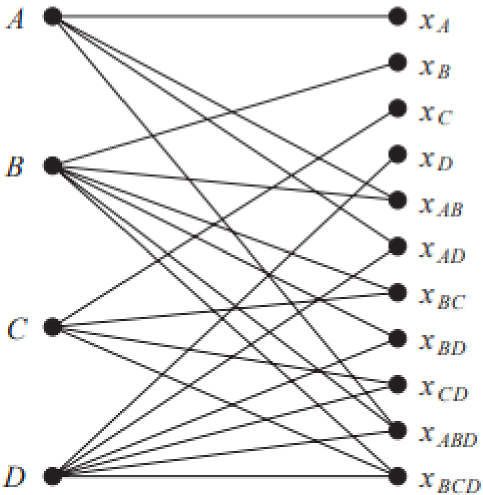
Therefore, the greedy algorithm is optimal on the complete bipartite graph! Break the graph down into Complete Bipartite subgraphs and then run greedy on those!

Definition (Cluster)

let $G = (X \cup P, E)$ be a bipartite graph and let $X' \subseteq X$ and $P' \subseteq P$. Then, $C = (X', P')$ is a *cluster for* G if G' induced by $X' \cup P'$ is complete bipartite. X(C) and P(C) denote X' and P'.

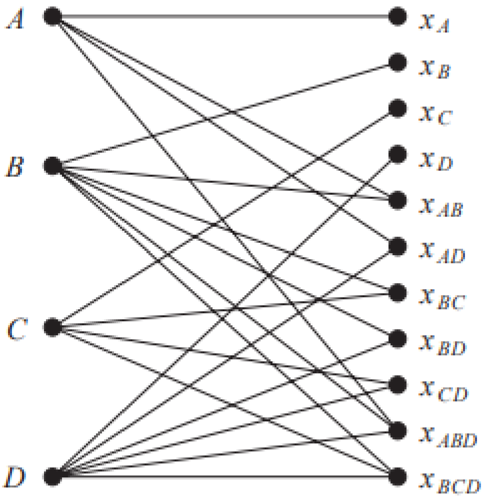
Definition (Neighbourhood of a Cluster)

Let C be a cluster then N(C) the *neighbourhood of* C is the set of vertices adjacent to any vertex in $X(C)$.


A
B
C
D
x_A
x_B
x_C
x_D
x_{AB}
x_{AD}
x_{BC}
x_{BD}
x_{CD}
x_{ABD}
x_{BCD}

Definition (Decomposition into Clusters)

A set $\mathcal{D}$ of clusters for a bipartite graph $G = (X \cup P, E)$ is a *decomposition into clusters of* G if every vertex of X belongs to one cluster of $\mathcal{D}$ and in P belongs to at most one cluster of $\mathcal{D}$


A
B
C
D
x A
x B
x C
x D
x AB
x AD
x BC
x BD
x CD
x ABD
x BCD

Cellular Network Channel Assignment Problem

Algorithm 3 Cluster Algorithm

1: **procedure** GreedyClusterAssignment
Initialize:
2: Create the Graph $G = (X \cup P, E)$
3: Decompose into clusters $\mathcal{D}$
To assign job $j = (w, x)$ **do:**
4: C $\leftarrow$ C$\in \mathcal{D}$ containing x
5: Assign j to $p \in C$ with lowest load

Competitive Ratio

The Cluster Algorithm is dependant on decomposition.
Define $r_w = max_{C \in \mathcal{D}}\{\frac{|N(C)|-1}{|P(C)|}\}$

Theorem

For any set of processors P and any set of task types $\mathcal{T}$ and for any decomposition into clusters $\mathcal{D}$ of the associated bipartite graph, the cluster algorithm is $(1 + r_w)$ competitive.

Cellular Network Channel Assignment Problem

Let p_i be the processor that reaches maximum load l during the algorithm. Let C_j be the cluster containing p_i. Consider an iteration where job j with weight w is assigned to p_i such that it reaches maximum load.

Then immediately before j arrives $\textbf{LOAD}(p_i) = l - w$

Because the algorithm assigns j to $min_{p \in C_k} \textbf{LOAD}(p)$, $l - w$ is a lowerbound on the weight of any machine. Therefore the total weight of jobs assigned to C_k is at-least $\mid P(C_k) \mid (l - w) + w$.

Proof of Competitive Ratio

Since those jobs can only be assigned to $N(C_k)$ and $w \leq \mathbf{OPT}$.

$$\begin{aligned}(l-w)\cdot \mid P(C_k)\mid + w &\leq \text{Total weight of jobs assigned to } P(C_k) \\ &\leq \text{Total weight of jobs assignable to } N(C_k) \\ &\leq \mid N(C_k)\mid \cdot \mathbf{OPT}\end{aligned}$$

Therefore,

$$\begin{aligned}\frac{l}{\mathbf{OPT}} = \frac{l-w}{\mathbf{OPT}} + \frac{w}{\mathbf{OPT}} &\leq \frac{\mid N(C_k)\mid}{\mid P(C_k)\mid} + \frac{w}{\mathbf{OPT}}\left(1 - \frac{1}{\mid P(C_k)\mid}\right) \\ &\leq 1 + \frac{\mid N(C_k)\mid}{\mid P(C_k)\mid} - \frac{1}{\mid P(C_k)\mid} \\ &= 1 + r_w\end{aligned}$$

Cellular Network Channel Assignment Problem

Theorem

For unit weight jobs the competitive ratio is $\max_{C \in \mathcal{D}} \frac{|N(C)|}{|P(C)|}$.

Cellular Network Channel Assignment Problem

Theorem

This is a tight bound on the performance.

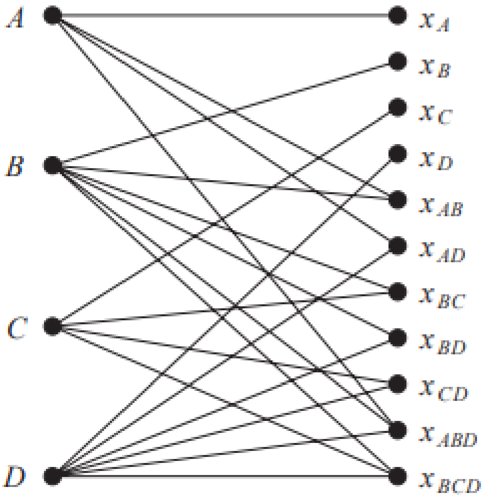
1. Create $(\mid N(C_k) \mid -1) \cdot \mathbf{OPT}$ jobs with weight 1 and type $x \in X(C_k)$
2. Cluster algorithm will assign at-least $\frac{(|N(C_k)|-1) \cdot \mathbf{OPT}}{|P(C_k)|} - 1$ weight to each $p \in P(C_k)$
3. Create A job of weight **OPT** and type $x \in X(C_k)$
4. Some processor in C_k has load at-least $\frac{(|N(C_k)|-1) \cdot \mathbf{OPT}}{|P(C_k)|} - 1 + \mathbf{OPT}$
5. Optimal off-line solution schedules **OPT** weight on each $p \in N(C_k)$

Cellular Network Channel Assignment Problem

Create an associated Bipartite graph as follows:

$$
\begin{aligned}
G =& (X \cup P, E) \\
P =& \text{The set of Base Stations} \\
X =& \{x_A \mid \text{A is a base station}\} \\
& \cup \{x_{AB} \mid \text{A and B are two intersecting base stations}\} \\
& \cup \{x_{ABC} \mid \text{A,B,C are three intersecting base stations}\} \\
E =& \{(x_A, A) \mid x_A \in X\} \\
& \cup \{(x_{AB}, y) \mid x_{AB} \in X, y \in \{A, B\}\} \\
& \cup \{(x_{ABC}, y) \mid x_{ABC} \in X, y \in \{A, B, C\}\}
\end{aligned}
$$

Cellular Network Channel Assignment Problem

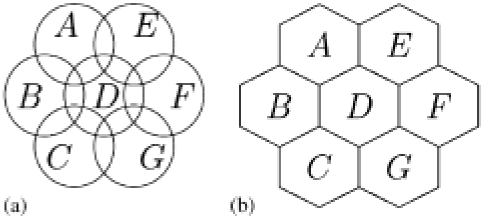


Hexagonal Grid Topology

Lemma

There exists a decomposition $\mathcal{D}$ into clusters such that, for any $C \in \mathcal{D}, |P(C) = 1|$ and $|N(C)| = 4$

For Cell D. The cluster containing D is $(\{x_D, x_{AD}, x_{BD}, x_{CD}, x_{ABD}, x_{BCD}\}, \{D\})$



Competitive Ratio for Cellular Networks

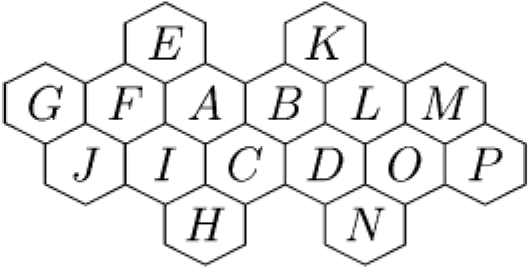
Theorem

The clustering algorithm is 4-competitive for unitary and arbitrary weights for the problem of channel assignment in cellular networks.

Greedy Lower Bound

Theorem

The greedy algorithm is at least 5-competitive, even in the case of unit weights



Theorem

Any online algorithm for the channel assignment problem in cellular networks is at-least 3 competitive, even for unit weights.

Yossi Azar et al. "On-Line Load Balancing of Temporary Tasks". In: *J. Algorithms* 22.1 (Jan. 1997), pp. 93–110. ISSN: 0196-6774. DOI: 10.1006/jagm.1995.0799. URL: https://doi.org/10.1006/jagm.1995.0799.

Amotz Bar-Noy, Ari Freund, and Joseph (Seffi) Naor. "New algorithms for related machines with temporary jobs". In: *Journal of Scheduling* 3.5 (2000), pp. 259–272. DOI: https://doi.org/10.1002/1099-1425(200009/10)3:5<259::AID-JOS47>3.0.CO;2-3. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/1099-1425. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/1099-1425%5C%28200009/10%5C%293%5C%3A5%5C%3C259%5C%3A%5C%3AAID-JOS47%5C%3E3.0.CO%5C%3B2-3.

Pilu Crescenzi, Giorgio Gambosi, and Paolo Penna. "On-line algorithms for the channel assignment problem in cellular networks". In: *Discrete Applied Mathematics* 137.3 (2004),

pp. 237–266. ISSN: 0166-218X. DOI:
https://doi.org/10.1016/S0166-218X(03)00341-X.
URL: https://www.sciencedirect.com/science/article/pii/S0166218X0300341X.