The Locality of Maximal Matching

Manuela Fischer
ETH Zurich
Locality
Locality
Locality
Locality
Locality
LOCAL Model Linial [FOCS’87]
LOCAL Model \textit{Linial [FOCS’87]}

standard synchronous message-passing model of distributed computing
LOCAL Model  \textit{Linial} [FOCS’87]

standard synchronous message-passing model of distributed computing
LOCAL Model \textit{Linial [FOCS’87]}

standard synchronous message-passing model of distributed computing

- undirected graph $G = (V, E)$, $n$ nodes, maximum degree $\Delta$
LOCAL Model \hspace{1em} Linial [FOCS’87]

**standard synchronous message-passing model of distributed computing**

- undirected graph $G = (V, E)$, $n$ nodes, maximum degree $\Delta$
LOCAL Model \textit{Linial [FOCS’87]}

standard synchronous message-passing model of distributed computing

• undirected graph $G = (V, E)$, 
n nodes, maximum degree $\Delta$
LOCAL Model  *Linial* [FOCS’87]

standard synchronous message-passing model of distributed computing

- undirected graph $G = (V, E)$, $n$ nodes, maximum degree $\Delta$
- each round, every node
  - receives messages (sent in previous round)
  - performs some computation
  - sends message to all its neighbors
LOCAL Model  

Linial [FOCS’87]

standard synchronous message-passing model of distributed computing

• undirected graph $G = (V, E)$, 
n nodes, maximum degree $\Delta$

• each round, every node
  • receives messages (sent in previous round)
  • performs some computation
  • sends message to all its neighbors

• unbounded message size
LOCAL Model  \textit{Linial [FOCS’87]}

standard synchronous message-passing model of distributed computing

- undirected graph $G = (V, E)$, $n$ nodes, maximum degree $\Delta$
- each round, every node
  - receives messages (sent in previous round)
  - performs some computation
  - sends message to all its neighbors
- unbounded message size
- unbounded computation
LOCAL Model  

Linial [FOCS’87]

standard synchronous message-passing model of distributed computing

• undirected graph $G = (V, E)$, $n$ nodes, maximum degree $\Delta$

• each round, every node
  • receives messages (sent in previous round)
  • performs some computation
  • sends message to all its neighbors

• unbounded message size

• unbounded computation

• Round Complexity: number of rounds to solve the problem
LOCAL Model \textit{Linial [FOCS’87]}

standard synchronous message-passing model of distributed computing

- undirected graph $G = (V, E)$, $n$ nodes, maximum degree $\Delta$
- each round, every node
  - receives messages (sent in previous round)
  - performs some computation
  - sends message to all its neighbors
- unbounded message size
- unbounded computation

**Round Complexity:**
number of rounds to solve the problem

Round complexity of a problem in the LOCAL model characterizes its locality
LOCAL Model  

Linial [FOCS’87]

standard synchronous message-passing model of distributed computing

• undirected graph $G = (V, E)$, $n$ nodes, maximum degree $\Delta$

• each round, every node
  • receives messages (sent in previous round)
  • performs some computation
  • sends message to all its neighbors

• unbounded message size
• unbounded computation

• **Round Complexity:**
  number of rounds to solve the problem

round complexity of a problem in the LOCAL model characterizes its locality
LOCAL Model  \( \text{Linial [FOCS’87]} \)

standard synchronous message-passing model of distributed computing

- undirected graph \( G = (V, E) \), n nodes, maximum degree \( \Delta \)
- each round, every node
  - receives messages (sent in previous round)
  - performs some computation
  - sends message to all its neighbors
- unbounded message size
- unbounded computation

\( \text{Round Complexity:} \)
number of rounds to solve the problem

round complexity of a problem in the LOCAL model characterizes its locality
**LOCAL Model**  *Linial [FOCS’87]*

**standard synchronous message-passing model of distributed computing**

- undirected graph $G = (V, E)$, $n$ nodes, maximum degree $\Delta$
- each round, every node
  - receives messages (sent in previous round)
  - performs some computation
  - sends message to all its neighbors
- unbounded message size
- unbounded computation

**Round Complexity:**
number of rounds to solve the problem

round complexity of a problem in the LOCAL model characterizes its locality
LOCAL Model  *Linial [FOCS’87]*

**standard synchronous message-passing model of distributed computing**

- undirected graph $G = (V, E)$, $n$ nodes, maximum degree $Δ$
- each round, every node
  - receives messages (sent in previous round)
  - performs some computation
  - sends message to all its neighbors
- unbounded message size
- unbounded computation

**Round Complexity:**
number of rounds to solve the problem

round complexity of a problem in the LOCAL model characterizes its locality
every problem is trivially solvable in $O(\text{diameter})$ rounds
Classic LOCAL Graph Problems
Classic LOCAL Graph Problems

Maximal Independent Set
Classic LOCAL Graph Problems

Maximal Independent Set

$(\Delta + 1)$-Vertex-Coloring
Classic LOCAL Graph Problems

Maximal Independent Set

$(\Delta + 1)$-Vertex-Coloring

Maximal Matching
**Classic LOCAL Graph Problems**

- **Maximal Independent Set**
- **(Δ + 1)-Vertex-Coloring**
- **Maximal Matching**
- **(2Δ − 1)-Edge-Coloring**
Classic LOCAL Graph Problems

- Maximal Independent Set
- $(\Delta + 1)$-Vertex-Coloring
- Maximal Matching
- $(2\Delta - 1)$-Edge-Coloring

Easy centralized problems: greedy solutions.
Maximal Matching
Maximal Matching
Maximal Matching
Maximal Matching

Matching:
set of non-incident edges
Maximal Matching

Matching:
set of non-incident edges

Maximal:
no edge can be added
Maximal Matching

Matching:
set of non-incident edges

Maximal:
no edge can be added

greedy property!
Centralized (Sequential) Algorithm
Centralized (Sequential) Algorithm
Centralized (Sequential) Algorithm
Centralized (Sequential) Algorithm
Centralized (Sequential) Algorithm
Centralized (Sequential) Algorithm
Centralized (Sequential) Algorithm
Centralized (Sequential) Algorithm
Centralized (Sequential) Algorithm
LOCAL Algorithm Mimicking Sequential Algorithm
LOCAL Algorithm Mimicking Sequential Algorithm
LOCAL Algorithm Mimicking Sequential Algorithm
LOCAL Algorithm Mimicking Sequential Algorithm
LOCAL Algorithm Mimicking Sequential Algorithm
LOCAL Algorithm Mimicking Sequential Algorithm
LOCAL Algorithm Mimicking Sequential Algorithm
LOCAL Algorithm Mimicking Sequential Algorithm
LOCAL Algorithm Mimicking Sequential Algorithm
LOCAL Algorithm Mimicking Sequential Algorithm
LOCAL Algorithm Mimicking Sequential Algorithm

can take $\Omega$ (diameter) rounds in worst case
LOCAL Algorithm Mimicking Sequential Algorithm

can take $\Omega(\text{diameter})$ rounds in worst case

Random Numbers:
$O(\log n)$ rounds w.h.p.
Luby [STOC’85]
F., Noever [SODA’18]
LOCAL Algorithm: Luby’s Randomized Algorithm
LOCAL Algorithm: Luby’s Randomized Algorithm
LOCAL Algorithm: Luby’s Randomized Algorithm
LOCAL Algorithm: Luby’s Randomized Algorithm
LOCAL Algorithm: Luby’s Randomized Algorithm
LOCAL Algorithm: Luby’s Randomized Algorithm

\[ \mathbb{E}[\#\text{removed edges per round}] \geq c|E_i| \]
LOCAL Algorithm: Luby’s Randomized Algorithm

\[ \mathbb{E}[\# \text{removed edges per round}] \geq c |E_i| \quad 0(\log n) \text{ rounds w.h.p.} \]
Our Result
Our Result

**Deterministic** $O(\log^2 \Delta \cdot \log n)$-round Maximal Matching
Our Result

deterministic $O(\log^2 \Delta \cdot \log n)$-round Maximal Matching

improving over
Our Result

\textbf{deterministic} \(O(\log^2 \Delta \cdot \log n)\)-round Maximal Matching improving over

\(O(\log^4 n)\)

Hańćkowiak, Karoński, Panconesi [SODA’98, PODC'99]
Our Result

**deterministic** $O(\log^2 \Delta \cdot \log n)$-round Maximal Matching improving over

$O(\log^4 n)$
Hańćkowiak, Karoński, Panconesi [SODA’98, PODC'99]

$O(\Delta + \log^* n)$
Panconesi, Rizzi [DIST’01]
## Overview of Results

### Maximal Matching
- Maximal Matching: $O(\log^2 \Delta \cdot \log n)$
- Randomized Maximal Matching: $O(\log^3 \log n + \log \Delta)$

### Approximate Matching
- $(2 + \varepsilon)$ - Approximate Maximum Matching: $O\left(\log^2 \Delta \cdot \log \frac{1}{\varepsilon} + \log^* n\right)$
- $(2 + \varepsilon)$ - Approximate Maximum Weighted Matching: $O\left(\log^2 \Delta \cdot \log \frac{1}{\varepsilon} + \log^* n\right)$
- $(2 + \varepsilon)$ - Approximate Maximum B-Matching: $O\left(\log^2 \Delta \cdot \log \frac{1}{\varepsilon} + \log^* n\right)$
- $(2 + \varepsilon)$ - Approximate Maximum Weighted B-Matching: $O\left(\log^2 \Delta \cdot \log \frac{1}{\varepsilon} + \log^* n\right)$
- $\varepsilon$ - Maximal Matching: $O\left(\log^2 \Delta \cdot \log \frac{1}{\varepsilon}\right)$
- $(2 + \varepsilon)$ - Approximate Minimum Edge Dominating Set: $O\left(\log^2 \Delta \cdot \log \frac{1}{\varepsilon}\right)$
| Constant - Approximate Bipartite Matching | $O(\log^2 \Delta)$ rounds |
Constant - Approximate Bipartite Matching

$O(\log^2 \Delta)$ rounds
| Constant - Approximate Bipartite Matching | $O(\log^2 \Delta)$ rounds |
I) 4 - Approximate Fractional Matching

$O(\log \Delta)$ rounds
I) 4 - Approximate Fractional Matching
   $O(\log \Delta)$ rounds

II) Rounding Fractional Bipartite Matching
   $O(\log^2 \Delta)$ rounds, $O(1)$ loss
1) 4-Approximate Fractional Matching $O(\log \Delta)$ rounds
1) 4-Approximate Fractional Matching

\[ \max_{\substack{x_e \in [0,1] \quad \text{for all } e \in E}} \sum_{e \in E} x_e \]

s.t.
\[ \sum_{e \in E(v)} x_e \leq 1 \quad \text{for all } v \in V \]

\[ x_e \in [0,1] \quad \text{for all } e \in E \]
I) 4-Approximate Fractional Matching

\[
\max \sum_{e \in E} x_e \quad \text{value of } \nu
\]

\[
\text{s.t. } \sum_{e \in E(v)} x_e \leq 1 \quad \text{for all } \nu \in V
\]

\[
x_e \in [0,1] \quad \text{for all } e \in E
\]

\[O(\log \Delta)\] rounds
I) 4-Approximate Fractional Matching \[ O(\log \Delta) \text{ rounds} \]

**Fractional Maximum Matching**

\[
\max \sum_{e \in E} x_e \quad \text{value of } v \\
\text{s.t. } \sum_{e \in E(v)} x_e \leq 1 \quad \text{for all } v \in V \\
x_e \in [0,1] \quad \text{for all } e \in E
\]

**LOCAL Greedy Algorithm**

\[
x_e = 2^{-\lceil \log \Delta \rceil} \quad \text{for all } e \in E
\]

repeat until all edges are blocked

- mark half-tight nodes
- block its edges
- double value of unblocked edges
I) 4-Approximate Fractional Matching

Fractional Maximum Matching

\[
\text{Fractional Maximum Matching} = \max \sum_{e \in E} x_e \text{ value of } v
\]

s.t.

\[
\sum_{e \in E(v)} x_e \leq 1 \quad \text{for all } v \in V
\]

\[
x_e \in [0,1] \quad \text{for all } e \in E
\]

LOCAL Greedy Algorithm

\[
x_e = 2^{-\lceil \log \Delta \rceil} \text{ for all } e \in E
\]

repeat until all edges are blocked

mark half-tight nodes

block its edges

double value of unblocked edges

\[O(\log \Delta)\] rounds
I) 4-Approximate Fractional Matching

### Fractional Maximum Matching

\[
\max \sum_{e \in E} x_e \quad \text{value of } v
\]

s.t.

\[
\sum_{e \in E(v)} x_e \leq 1 \quad \text{for all } v \in V
\]

\[
x_e \in [0,1] \quad \text{for all } e \in E
\]

### LOCAL Greedy Algorithm

\[x_e = 2^{-[\log \Delta]} \text{ for all } e \in E\]

repeat until all edges are blocked

mark half-tight nodes

block its edges

double value of unblocked edges

\[O(\log \Delta) \text{ rounds}\]
I) 4-Approximate Fractional Matching

\[ \text{Fractional Maximum Matching} \]

\[
\max \sum_{e \in E} x_e \quad \text{value of } v
\]

s.t. \[
\sum_{e \in E(v)} x_e \leq 1 \quad \text{for all } v \in V
\]

\[
x_e \in [0,1] \quad \text{for all } e \in E
\]

\[ v \text{ is half-tight if its value is } \geq \frac{1}{2} \]

\[ \text{LOCAL Greedy Algorithm} \]

\[
x_e = 2^{-\lceil \log \Delta \rceil} \text{ for all } e \in E
\]

repeat until all edges are blocked
mark half-tight nodes
block its edges
double value of unblocked edges

\[ O(\log \Delta) \text{ rounds} \]
I) 4-Approximate Fractional Matching

\[ \frac{1}{16} \]

Fractional Maximum Matching

\[ \max \sum_{e \in E} x_e \]

\[ \text{value of } v \]

s.t.

\[ \sum_{e \in \delta(v)} x_e \leq 1 \quad \text{for all } v \in V \]

\[ x_e \in [0,1] \quad \text{for all } e \in E \]

\( v \) is half-tight if its value is \( \geq \frac{1}{2} \)

LOCAL Greedy Algorithm

\[ x_e = 2^{-[\log \Delta]} \quad \text{for all } e \in E \]

repeat until all edges are blocked

mark half-tight nodes

block its edges

double value of unblocked edges

\( O(\log \Delta) \) rounds
I) 4-Approximate Fractional Matching

\[ \text{Fractional Maximum Matching} \]

\[
\begin{align*}
\max \sum_{e \in E} x_e & \quad \text{value of } v \\
\text{s.t.} \sum_{e \in E(v)} x_e & \leq 1 \quad \text{for all } v \in V \\
x_e & \in [0,1] \quad \text{for all } e \in E
\end{align*}
\]

\( v \) is half-tight if its value is \( \geq \frac{1}{2} \)

\[ \text{LOCAL Greedy Algorithm} \]

\[ x_e = 2^{-\lceil \log \Delta \rceil} \text{ for all } e \in E \]

repeat until all edges are blocked

mark half-tight nodes

block its edges

double value of unblocked edges
I) 4-Approximate Fractional Matching

Fractional Maximum Matching

\[
\max_{e \in E} \sum_{e \in E} x_e
\]

subject to:

\[
\sum_{e \in E(v)} x_e \leq 1 \quad \text{for all } v \in V
\]

\[
x_e \in [0,1] \quad \text{for all } e \in E
\]

\(v\) is half-tight if its value is \(\geq \frac{1}{2}\)

LOCAL Greedy Algorithm

\[
x_e = 2^{-\lceil \log \Delta \rceil} \quad \text{for all } e \in E
\]

repeat until all edges are blocked

- mark half-tight nodes
- block its edges
- double value of unblocked edges

\(O(\log \Delta)\) rounds
I) 4-Approximate Fractional Matching

**Fractional Maximum Matching**

\[
\max \sum_{e \in E} x_e
\]

s.t.

\[
\sum_{e \in E(v)} x_e \leq 1 \quad \text{for all } v \in V
\]

\[
x_e \in [0,1] \quad \text{for all } e \in E
\]

\[v \text{ is half-tight if its value is } \geq \frac{1}{2}\]

**LOCAL Greedy Algorithm**

\[x_e = 2^{-\lceil \log \Delta \rceil} \text{ for all } e \in E\]

repeat until all edges are blocked

- mark half-tight nodes
- block its edges
- double value of unblocked edges

\[O(\log \Delta) \text{ rounds}\]
1) 4-Approximate Fractional Matching

**Fractional Maximum Matching**

\[
\max \sum_{e \in E} x_e \quad \text{value of } v
\]

s.t.

\[
\sum_{e \in E(v)} x_e \leq 1 \quad \text{for all } v \in V
\]

\[
x_e \in [0,1] \quad \text{for all } e \in E
\]

\[v \text{ is half-tight if its value is } \geq \frac{1}{2}\]

**LOCAL Greedy Algorithm**

\[x_e = 2^{-\lfloor \log \Delta \rfloor} \text{ for all } e \in E\]

repeat until all edges are blocked

mark half-tight nodes

block its edges

double value of unblocked edges

\[O(\log \Delta) \text{ rounds}\]
1) 4-Approximate Fractional Matching

$O(\log \Delta)$ rounds

**Fractional Maximum Matching**

$$\max \sum_{e \in E} x_e$$

value of $v$

s.t. $$\sum_{e \in E(v)} x_e \leq 1 \quad \text{for all } v \in V$$

$$x_e \in [0,1] \quad \text{for all } e \in E$$

$v$ is half-tight if its value is $\geq \frac{1}{2}$

**LOCAL Greedy Algorithm**

$$x_e = 2^{-[\log \Delta]} \text{ for all } e \in E$$

repeat until all edges are blocked

mark half-tight nodes

block its edges

double value of unblocked edges
I) 4-Approximate Fractional Matching

\[ \text{Fractional Maximum Matching} \]
\[
\max \sum_{e \in E} x_e \\
\text{s.t.} \sum_{e \in E(v)} x_e \leq 1 \quad \text{for all } v \in V \\
x_e \in [0,1] \quad \text{for all } e \in E
\]

\( v \) is half-tight if its value is \( \geq \frac{1}{2} \)

\[ \text{LOCAL Greedy Algorithm} \]
\[ x_e = 2^{-\lceil \log \Delta \rceil} \quad \text{for all } e \in E \]
repeat until all edges are blocked
mark half-tight nodes
block its edges
double value of unblocked edges

\( O(\log \Delta) \) rounds
I) 4-Approximate Fractional Matching

Fractional Maximum Matching

\[
\max \sum_{e \in E} x_e
\]

subject to

\[
\sum_{e \in E(v)} x_e \leq 1 \quad \text{for all } v \in V
\]

\[
x_e \in [0,1] \quad \text{for all } e \in E
\]

\(v\) is half-tight if its value is \(\geq \frac{1}{2}\)

LOCAL Greedy Algorithm

\[
x_e = 2^{-\lceil \log \Delta \rceil} \quad \text{for all } e \in E
\]

repeat until all edges are blocked

mark half-tight nodes

block its edges

double value of unblocked edges
I) 4-Approximate Fractional Matching
   $O(\log \Delta)$ rounds

II) Rounding Fractional Bipartite Matching
    $O(\log^2 \Delta)$ rounds, $O(1)$ loss
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss
II) Rounding Fractional Bipartite Matching

\( O(\log^2 \Delta) \) rounds, \( O(1) \) loss
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

Direct Rounding

$\Omega \left( \frac{1}{\Delta} \right)$

Gradual Rounding

$O(\log \Delta)$ iterations
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

**Factor-2-Rounding**

$\geq \frac{1}{d}$ $\Rightarrow$ $\geq \frac{2}{d}$

Direct Rounding $\Omega\left(\frac{1}{\Delta}\right)$ $1$

Gradual Rounding $0(\log \Delta)$ iterations
II) Rounding Fractional Bipartite Matching

\[ O(\log^2 \Delta) \text{ rounds, } O(1) \text{ loss} \]

**Factor-2-Rounding**

\[ \geq \frac{1}{d} \quad \Rightarrow \quad \geq \frac{2}{d} \]

using Locally Balanced Splitting, inspired by

Hańćkowiak, Karoński, Panconesi [SODA’98, PODC’99]

Direct Rounding

\[ \Omega\left(\frac{1}{\Delta}\right) \]

Gradual Rounding

\[ 0(\log \Delta) \text{ iterations} \]
| II) Rounding Fractional Bipartite Matching | $O(\log^2 \Delta)$ rounds, $O(1)$ loss |

Iterated Factor-2-Rounding using Locally Balanced Splitting
II) Rounding Fractional Bipartite Matching

Iterated Factor-2-Rounding using Locally Balanced Splitting

Locally Balanced Splitting:
2-edge-coloring so that
every node roughly balanced

$O(\log^2 \Delta)$ rounds, $O(1)$ loss
Iterated Factor-2-Rounding using Locally Balanced Splitting

**Locally Balanced Splitting:**
2-edge-coloring so that every node roughly balanced

II) Rounding Fractional Bipartite Matching

\[ O(\log^2 \Delta) \text{ rounds, } O(1) \text{ loss} \]
II) Rounding Fractional Bipartite Matching

Iterated Factor-2-Rounding using Locally Balanced Splitting

**Locally Balanced Splitting:**
2-edge-coloring so that every node roughly balanced

$O(\log^2 \Delta)$ rounds, $O(1)$ loss
Iterated Factor-2-Rounding using Locally Balanced Splitting

Locally Balanced Splitting:
2-edge-coloring so that every node roughly balanced

Iterated Factor-2-Rounding
for $i = \lceil \log \Delta \rceil, \ldots, 1$

$E_i = \{ e \in E : x_e = 2^{-i} \}$

splitting of $E_i$ into

increase to $2^{-i+1}$

decrease to 0

II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

Iterated Factor-2-Rounding using Locally Balanced Splitting

**Locally Balanced Splitting:**
2-edge-coloring so that every node roughly balanced

**Iterated Factor-2-Rounding**

for $i = \lceil \log \Delta \rceil, \ldots, 1$

$E_i = \{ e \in E : x_e = 2^{-i} \}$

splitting of $E_i$ into

increase to $2^{-i+1}$

decrease to 0
II) Rounding Fractional Bipartite Matching

Iterated Factor-2-Rounding using Locally Balanced Splitting

Locally Balanced Splitting:
2-edge-coloring so that every node roughly balanced

Iterated Factor-2-Rounding
for $i = \lfloor \log \Delta \rfloor, \ldots, 1$

$$E_i = \{ e \in E : x_e = 2^{-i} \}$$

splitting of $E_i$ into
increase to $2^{-(i+1)}$
decrease to 0

$O(\log^2 \Delta)$ rounds, $O(1)$ loss
II) Rounding Fractional Bipartite Matching

*Iterated Factor-2-Rounding using Locally Balanced Splitting*

**Locally Balanced Splitting:**
2-edge-coloring so that every node roughly balanced

---

**Iterated Factor-2-Rounding**

for $i = \lceil \log \Delta \rceil, \ldots, 1$

$$E_i = \{ e \in E : x_e = 2^{-i} \}$$

splitting of $E_i$ into increase to $2^{-i+1}$
decrease to 0

---

In case of **perfect locally balanced splitting**: no constraint violated & no loss in total value (i.e., **perfect rouding**)

---

$\mathcal{O}(\log^2 \Delta)$ rounds, $\mathcal{O}(1)$ loss
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

Iterated Factor-2-Rounding using Locally Balanced Splitting

Locally Balanced Splitting:
2-edge-coloring so that every node roughly balanced

Iterated Factor-2-Rounding
for $i = \lfloor \log \Delta \rfloor, \ldots, 1$

$E_i = \{ e \in E : x_e = 2^{-i} \}$

splitting of $E_i$ into
increase to $2^{-i+1}$
decrease to 0

In case of **perfect** locally balanced splitting:
no constraint violated & no loss in total value
(i.e., **perfect rounding**)
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

Iterated Factor-2-Rounding using Locally Balanced Splitting

**Locally Balanced Splitting:**
2-edge-coloring so that every node roughly balanced

**Iterated Factor-2-Rounding**
for $i = [\log \Delta], \ldots, 1$

$E_i = \{ e \in E : x_e = 2^{-i} \}$

splitting of $E_i$ into

increase to $2^{-i+1}$

decrease to 0

In case of **perfect** locally balanced **splitting:**
no constraint violated & no loss in total value
(i.e., **perfect rounding**)
Iterated Factor-2-Rounding using Locally Balanced Splitting

Locally Balanced Splitting:
2-edge-coloring so that every node roughly balanced

Iterated Factor-2-Rounding
for $i = \lceil \log \Delta \rceil, \ldots, 1$

$E_i = \{ e \in E : x_e = 2^{-i} \}$

splitting of $E_i$ into

increase to $2^{-i+1}$

decrease to 0

In case of **perfect** locally balanced splitting: no constraint violated & no loss in total value (i.e., **perfect rounding**)
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

Iterated Factor-2-Rounding using Locally Balanced Splitting

Locally Balanced Splitting:
2-edge-coloring so that every node roughly balanced

Iterated Factor-2-Rounding

for $i = \lfloor \log \Delta \rfloor, \ldots, 1$

$E_i = \{ e \in E : x_e = 2^{-i} \}$

splitting of $E_i$ into

increase to $2^{-i+1}$

decrease to 0

In case of perfect locally balanced splitting:
no constraint violated & no loss in total value
(i.e., perfect rounding)
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

Iterated Factor-2-Rounding using Locally Balanced Splitting

**Locally Balanced Splitting:**
2-edge-coloring so that every node roughly balanced

**Iterated Factor-2-Rounding**
for $i = \lceil \log \Delta \rceil, \ldots, 1$

$E_i = \{ e \in E : x_e = 2^{-i} \}$

splitting of $E_i$ into

increase to $2^{-i+1}$

decrease to 0

In case of **perfect** locally balanced **splitting**:
no constraint violated & no loss in total value
(i.e., **perfect rounding**)
Iterated Factor-2-Rounding using Locally Balanced Splitting

Locally Balanced Splitting:
2-edge-coloring so that every node roughly balanced

Iterated Factor-2-Rounding
for $i = \lfloor \log \Delta \rfloor, \ldots, 1$

$E_i = \{ e \in E : x_e = 2^{-i} \}$

splitting of $E_i$ into

increase to $2^{-i+1}$
decrease to 0

In case of **perfect** locally balanced splitting:
no constraint violated & no loss in total value
(i.e., **perfect rouding**)
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

Iterated Factor-2-Rounding using Locally Balanced Splitting

**Locally Balanced Splitting:**
2-edge-coloring so that every node roughly balanced

**Iterated Factor-2-Rounding**
for $i = \lceil \log \Delta \rceil, \ldots, 1$

$$E_i = \{ e \in E : x_e = 2^{-i} \}$$

splitting of $E_i$ into

increase to $2^{-i+1}$

decrease to 0

In case of **perfect** locally balanced **splitting:**
no constraint violated & no loss in total value
(i.e., **perfect rouding**)
Iterated Factor-2-Rounding using Locally Balanced Splitting

**Locally Balanced Splitting:**
2-edge-coloring so that every node roughly balanced

**Iterated Factor-2-Rounding**

for \( i = \lceil \log \Delta \rceil, \ldots, 1 \)

\[
E_i = \{ e \in E : x_e = 2^{-i} \}
\]

splitting of \( E_i \) into

increase to \( 2^{-i+1} \)

decrease to 0

In case of **perfect** locally balanced splitting: no constraint violated & no loss in total value (i.e., **perfect rounding**)

II) Rounding Fractional Bipartite Matching

\( O(\log^2 \Delta) \) rounds, \( O(1) \) loss
II) Rounding Fractional Bipartite Matching

Iterated Factor-2-Rounding using Locally Balanced Splitting

Locally Balanced Splitting:
2-edge-coloring so that every node roughly balanced

Iterated Factor-2-Rounding
for $i = [\log \Delta], \ldots, 1$
$E_i = \{ e \in E : x_e = 2^{-i} \}$
splitting of $E_i$ into
increase to $2^{-i+1}$
decrease to 0

In case of perfect locally balanced splitting: no constraint violated & no loss in total value (i.e., perfect rounding)

$O(\log^2 \Delta)$ rounds, $O(1)$ loss
Iterated Factor-2-Rounding using Locally Balanced Splitting

**Locally Balanced Splitting:**
2-edge-coloring so that every node roughly balanced

**Iterated Factor-2-Rounding**

for $i = \lceil \log \Delta \rceil, \ldots, 1$

$E_i = \{ e \in E : x_e = 2^{-i} \}$

splitting of $E_i$ into

increase $\square$ to $2^{-i+1}$

decrease $\square$ to 0

In case of **perfect** locally balanced **splitting:**
no constraint violated & no loss in total value
(i.e., **perfect rouding**)

II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss
Iterated Factor-2-Rounding using Locally Balanced Splitting

**Locally Balanced Splitting:**
2-edge-coloring so that every node roughly balanced

**Iterated Factor-2-Rounding**
for $i = \lceil \log \Delta \rceil, \ldots, 1$

$E_i = \{ e \in E : x_e = 2^{-i} \}$

splitting of $E_i$ into

increase to $2^{-i+1}$

decrease to 0

In case of **perfect** locally balanced **splitting:**
no constraint violated & no loss in total value
(i.e., perfect rounding)
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

Iterated Factor-2-Rounding using Locally Balanced Splitting

Locally Balanced Splitting:
2-edge-coloring so that every node roughly balanced

Iterated Factor-2-Rounding
for $i = [\log \Delta], \ldots, 1$

$E_i = \{e \in E : x_e = 2^{-i}\}$
splitting of $E_i$ into
increase $x_e$ to $2^{-i+1}$
decrease $x_e$ to 0

In case of perfect locally balanced splitting:
no constraint violated & no loss in total value
(i.e., perfect rounding)
| II) Rounding Fractional Bipartite Matching | $O(\log^2 \Delta)$ rounds, $O(1)$ loss |
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

Perfect Splitting not possible in case of...
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

Perfect Splitting not possible in case of...

... odd cycles
II) Rounding Fractional Bipartite Matching

Perfect Splitting not possible in case of...

... odd cycles

bipartite graph!

$O(\log^2 \Delta)$ rounds, $O(1)$ loss
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

Perfect Splitting not possible in case of...

- odd cycles
- odd-degree vertices

bipartite graph!
II) Rounding Fractional Bipartite Matching

\[ O(\log^2 \Delta) \text{ rounds, } O(1) \text{ loss} \]

Perfect Splitting not possible in case of...

... odd cycles

... odd-degree vertices

bipartite graph!

small technicality.
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

Perfect Splitting not possible in case of...

... odd cycles

bipartite graph!

... odd-degree vertices

small technicality.

Suppose that bipartite and even degree!
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

**Sequential Perfect Splitting**

Repeat until all edges colored
pick arbitrary cycle
alternate

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

Sequential Perfect Splitting*

Repeat until all edges colored
pick arbitrary cycle
alternate

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

**Sequential Perfect Splitting**

Repeat until all edges colored
pick arbitrary cycle
alternate

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

\[ O(\log^2 \Delta) \text{ rounds, } O(1) \text{ loss} \]

Sequential Perfect Splitting*

Repeat until all edges colored
pick arbitrary cycle
alternate

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

Sequential Perfect Splitting*
Repeat until all edges colored
pick arbitrary cycle
alternate

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

Sequential Perfect Splitting*
Repeat until all edges colored
pick arbitrary cycle
alternate

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

**Sequential Perfect Splitting**
Repeat until all edges colored
pick arbitrary cycle
alternate

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

* Sequential Perfect Splitting*

Repeat until all edges colored
pick arbitrary cycle
alternate

* LOCAL Almost-Perfect Splitting*

Decompose into edge-disjoint cycles
In parallel, for all cycles
A) **Short cycles** of length $O(\log \Delta)$
   alternate
B) **Long cycles**
   chop at length $\Theta(\log \Delta)$
   set boundary to 0
   alternate in between

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

Sequential Perfect Splitting*
Repeat until all edges colored
pick arbitrary cycle
alternate

LOCAL Almost-Perfect Splitting*
Decompose into edge-disjoint cycles
In parallel, for all cycles
A) Short cycles of length $O(\log \Delta)$
   alternate
B) Long cycles
   chop at length $\Theta(\log \Delta)$
   set boundary to 0
   alternate
   in between

*$\text{bipartite and even degree!}$
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

Sequential Perfect Splitting*
Repeat until all edges colored
pick arbitrary cycle
alternate

LOCAL Almost-Perfect Splitting*
Decompose into edge-disjoint cycles
In parallel, for all cycles
A) Short cycles of length $O(\log \Delta)$
   alternate
B) Long cycles
   chop at length $\Theta(\log \Delta)$
   set boundary to 0
   alternate in between

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

Sequential Perfect Splitting*
Repeat until all edges colored
pick arbitrary cycle
alternate

LOCAL Almost-Perfect Splitting*
Decompose into edge-disjoint cycles
In parallel, for all cycles
A) *Short cycles* of length $O(\log \Delta)$
   alternate
B) *Long cycles*
   chop at length $\Theta(\log \Delta)$
   set boundary to 0
   alternate in between

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

**Sequential Perfect Splitting**
Repeat until all edges colored
pick arbitrary cycle
alternate

**LOCAL Almost-Perfect Splitting**
Decompose into edge-disjoint cycles
In parallel, for all cycles

A) **Short cycles** of length $O(\log \Delta)$
   alternate

B) **Long cycles**
   chop at length $\Theta(\log \Delta)$
   set boundary to 0
   alternate in between

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

Sequential Perfect Splitting*
Repeat until all edges colored
pick arbitrary cycle
alternate

LOCAL Almost-Perfect Splitting*
Decompose into edge-disjoint cycles
In parallel, for all cycles
A) **Short cycles** of length $O(\log \Delta)$
   alternate
B) **Long cycles**
   chop at length $\Theta(\log \Delta)$
   set boundary to 0
   alternate

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

$O(\log^2 \Delta)$ rounds, $O(1)$ loss

**Sequential Perfect Splitting**
Repeat until all edges colored
pick arbitrary cycle
alternate

**LOCAL Almost-Perfect Splitting**
Decompose into edge-disjoint cycles
In parallel, for all cycles
A) **Short cycles** of length $O(\log \Delta)$
   alternate
B) **Long cycles**
   chop at length $\Theta(\log \Delta)$
   set boundary to 0
   alternate in between

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

- **Sequential Perfect Splitting**
  - Repeat until all edges colored
  - Pick arbitrary cycle
  - Alternate

- **LOCAL Almost-Perfect Splitting**
  - Decompose into edge-disjoint cycles
  - In parallel, for all cycles
    - **A)** Short cycles of length $O(\log \Delta)$
      - Alternate
    - **B)** Long cycles
      - Chop at length $\Theta(\log \Delta)$
      - Set boundary to 0
      - Alternate

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

Sequential Perfect Splitting*
Repeat until all edges colored
pick arbitrary cycle
alternate

LOCAL Almost-Perfect Splitting*
Decompose into edge-disjoint cycles
In parallel, for all cycles
A) Short cycles of length $O(\log \Delta)$
   alternate
B) Long cycles
   chop at length $\Theta(\log \Delta)$
   set boundary to 0
   alternate in between

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

**Sequential Perfect Splitting**
Repeat until all edges colored
pick arbitrary cycle
alternate

**LOCAL Almost-Perfect Splitting**
Decompose into edge-disjoint cycles
In parallel, for all cycles
A) **Short cycles** of length $O(\log \Delta)$
   alternate
B) **Long cycles**
   chop at length $\Theta(\log \Delta)$
   set boundary to 0
   alternate

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

\[ O(\log^2 \Delta) \text{ rounds, } O(1) \text{ loss} \]

**Sequential Perfect Splitting***
Repeat until all edges colored
pick arbitrary cycle
alternate

**LOCAL Almost-Perfect Splitting***
Decompose into edge-disjoint cycles
In parallel, for all cycles
A) **Short cycles** of length \( O(\log \Delta) \)
alternate

B) **Long cycles**
chop at length \( \Theta(\log \Delta) \)
set boundary to 0
alternate

\[ \Theta \left( \frac{1}{\log \Delta} \right) \text{ loss} \]
in between

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

---

**Sequential Perfect Splitting**
Repeat until all edges colored
pick arbitrary cycle
alternate

**LOCAL Almost-Perfect Splitting**
Decompose into edge-disjoint cycles
In parallel, for all cycles
A) **Short cycles** of length $O(\log \Delta)$
   alternate
B) **Long cycles**
   chop at length $\Theta(\log \Delta)$
   set boundary to 0
   alternate

*$\Theta\left(\frac{1}{\log \Delta}\right)$ loss

* by Hańckowiak, Karoński, Panconesi [SODA’98,PODC’99] in $O(\log \Delta)$

* bipartite and even degree!
II) Rounding Fractional Bipartite Matching

*Sequential Perfect Splitting*
Repeat until all edges colored
pick arbitrary cycle
alternate

*LOCAL Almost-Perfect Splitting*
Decompose into edge-disjoint cycles
In parallel, for all cycles
A) **Short cycles** of length $O(\log \Delta)$
   alternate
B) **Long cycles**
   chop at length $\Theta(\log \Delta)$
   set boundary to 0
   alternate

Over all $O(\log \Delta)$ rounding iterations, total loss still constant!

* by Hańćkowiak, Karoński, Panconesi [SODA’98,PODC’99] in $O(\log \Delta)$

* bipartite and even degree!
I) 4-Approximate Fractional Matching
   $O(\log \Delta)$ rounds

II) Rounding Fractional Bipartite Matching
   $O(\log^2 \Delta)$ rounds, $O(1)$ loss
| Constant - Approximate Bipartite Matching | $O(\log^2 \Delta)$ rounds |
Constant - Approximate Bipartite Matching \[ O(\log^2 \Delta) \] rounds
Constant - Approximate Bipartite Matching

$O(\log^2 \Delta)$ rounds
Constant - Approximate Bipartite Matching

$O(\log^2 \Delta)$ rounds
Constant - Approximate Bipartite Matching

$O(\log^2 \Delta)$ rounds

![Diagram showing the process of approximate bipartite matching with $O(\log^2 \Delta)$ rounds.](image-url)
Constant - Approximate Bipartite Matching

\[ O(\log^2 \Delta) \text{ rounds} \]
Constant - Approximate Bipartite Matching

$O(\log^2 \Delta)$ rounds

Maximal Matching in Degree-2-Graph

$O(1)$ rounds, $O(1)$-factor loss

Panconesi, Rizzi [DIST'01]

Constant-Approximate Bipartite Matching

$O(\log^2 \Delta)$ rounds
<table>
<thead>
<tr>
<th>Constant - Approximate Matching</th>
<th>( O(\log^2 \Delta) ) rounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximal</td>
<td>( O(\log^2 \Delta \cdot \log n) )</td>
</tr>
</tbody>
</table>
Constant - Approximate Matching

Maximal

$O \left( \log^2 \Delta \right)$ rounds

$O \left( \log^2 \Delta \cdot \log n \right)$
Constant - Approximate Matching

Maximal

$O(\log^2 \Delta \cdot \log n)$
Constant - Approximate Matching

Maximal

$O(\log^2 \Delta) \text{ rounds}$

$O(\log^2 \Delta \cdot \log n)$
Constant - Approximate Matching

Maximal

$O(\log^2 \Delta \cdot \log n)$
Constant - Approximate Matching

Maximal

$O(\log^2 \Delta \cdot \log n)$
Constant - Approximate Matching

Maximal

\( O(\log^2 \Delta \cdot \log n) \)
Constant - Approximate Matching

Maximal

$O(\log^2 \Delta \cdot \log n)$

$O(\log^2 \Delta)$ rounds
Constant - Approximate Matching

Maximal

$O(\log^2 \Delta \cdot \log n)$
Constant - Approximate Matching

Maximal

\[ O(\log^2 \Delta \cdot \log n) \]

maximum matching size in remainder graph decreases by constant factor
Constant - Approximate Matching

Maximal

\[ O(\log^2 \Delta \cdot \log n) \]

maximum matching size in remainder graph decreases by constant factor

after \( O(\log n) \) iterations, maximum matching size is 0, hence graph empty
Open Question: $\Theta(\log \Delta \cdot \log n)$?

What is Locality of Maximal Matching?
Open Question: $O(\log \Delta \cdot \log n)$?

What is Locality of Maximal Matching?