PRAMs over integers do not compute maxflow efficiently

Thomas Seiller
CNRS, LIPN (Paris 13 Univ.)

DICE-FOPARA 2019
April 6-7th, 2019
Context and the main result
Parallel Computation

PRAMs provide the basic model for parallel computation.

- Unbounded collection of numbered RAM processors $P_0$, $P_1$, $P_2$, etc.
- Each $P_i$ has its own (unbounded) local memory (registers) and knows its index $i$.
- Input consists of $n$ items stored in (usually the first) $n$ shared memory cells.
- Output consists of $m$ items stored in $m$ shared memory cells.
- Instructions execute in 3-phase cycles: Read (if any) from a shared memory cell, Local computation (if any). Write (if any) to a shared memory cell.
- Processors execute these 3-phase cycles synchronously, and special assumptions have to be made about R-R and W-W shared memory access conflicts.
The big question

Definition

The class NC is defined as the problems computable by a family of PRAMs $M_i$ with input of size $i$, such that each $M_i$ is of polynomial width (i.e. number of processors) and polylogarithmic depth.

The question "Is PTIME equal to NC?" corresponds to know if parallelisation always leads to substantial gain in running time (for feasible problems).

Definition

$maxflow$ is the following problem: given a weighted graph with both a source $s$ and a sink $t$, and a value $n$, decide whether there exists a flow of total value $n$ from $s$ to $t$.

Theorem (Lengauer, Wagner 1990)

$maxflow$ is PTIME-complete.
PRAMs over integers do not compute maxflow efficiently.
Mulmuley’s result

PRAMs without bit operations (PRAM\(^{w/o}\)s) are defined as a model of \textit{algebraic} PRAMs. I.e. registers contain integers. Processors are allowed basic arithmetic operations +, ×, −, and comparisons =, <, ≤.

**Theorem (Mulmuley 1999)**

\begin{quote}
Let \( G \) be a PRAM\(^{w/o}\) with \( 2^{O((\log N)^c)} \) processors, where \( N \) is the length of the inputs and \( c \) any positive integer.
Then \( G \) does not decide \textit{maxflow} in \( O((\log N)^c) \) steps.
\end{quote}

- This is considered as one of the strongest lower bound results, as the model can be shown to compute several non-trivial problems in NC.
- It lead Mulmuley to introduce the Geometric Complexity Theory (GCT) programme aiming at a proof of \textit{PTIME} ≠\textit{NPTIME} using techniques from algebraic geometry.
- However, this result has not been improved since, and GCT has recently been the subject of some negative results (Burgisser, Ikenmeyer, Panova 2016)
A small step forward: our main result

**Z-PRAMs** are defined as a model of *algebraic* PRAMs. I.e. registers contain integers. Processors are allowed basic arithmetic operations $+$, $\times$, $-$ and $\div$, and comparisons $=,<,\leq$.

Theorem (Pellissier & Seiller)

*Let $G$ be a Z-PRAM with $2^{O((\log N)^c)}$ processors, where $N$ is the length of the inputs and $c$ any positive integer. Then $G$ does not decide maxflow in $O((\log N)^c)$ steps.*

Defining $\text{NC}_Z$ as the class of problems decided by a family of Z-PRAMs $M_i$ with input of size $i$, such that each $M_i$ is of polynomial width (i.e. number of processors) and polylogarithmic depth.

$\text{NC}_Z \neq \text{PTIME}$
The method
What’s a graphing?

- (Multi)Graph = Collection of edges.
- Graphing = Collection of realised edges.
- Replace vertices by subspaces of the underlying space $\mathbf{X}$.
- Decide how (i.e. pick an element of $\mathbf{M}$ a monoid of $\mathbf{X}$ endomorphisms) the edges map sources to targets.

Example:

- $\mathbf{X} = \{(s_i,j) | (s_i) \in \{0, 1\}_0, j \in \mathbb{Z}\}$
- $\mathbf{M} \leq \mathbf{X} \rightarrow \mathbf{X}$ generated by $((s_i),j) \mapsto ((s_i),j+1)$, $((s_i),j) \mapsto ((s_i),j-1)$, and $w_s : ((s_i),j) \mapsto ((\hat{s}_i),j)$ with $\hat{s}_j = s$ and $\hat{s}_i = s_i$ for $i \neq j$. 

\[ \alpha(a) : x \mapsto 5 - x \]
\[ \alpha(b) : x \mapsto (x - 1)^2 + 2 \]

```
[0,1]        [1,2]        [3,4]        [4,5]
```

\[ \alpha(a) \]
\[ \alpha(b) \]

T. Seiller, CNRS
PRAMs over integers do not compute maxflow efficiently
April 6th, 2019
A bit more on graphings

Graphings were introduced by Adams in the context of ergodic theory. They also appear as certain limits of graphs (cf. Aldous-Lyons conjecture).

**Definition**

Given an AMC $\alpha : M \sim X$, an $\alpha$-graphing is a collection of pairs $(S, m)$ where $S \subset X$ and $m \in M$.

One can consider restrictions of graphings:

- Discrete space: multigraphs;
- Deterministic Graphings: Dynamical Systems
- Probabilistic Graphings: Markov processes

-PRAMS over integers do not compute maxflow efficiently
Dynamic complexity

Definition
An abstract model of computation (AMC) is defined as a monoid action \( \alpha : M \curvearrowright X \) of a monoid \( M \) on a space \( X \). I.e. \( \alpha : M \rightarrow \mathcal{M}(X \rightarrow X) \).

Definition
An abstract program is a \( \alpha(M) \)-graphing.

The set of \( \alpha(M) \)-graphings is typed by (at least) multiplicative-additive linear logic. The type \( \textbf{Nat} \rightarrow \textbf{Bool} \) defines a complexity class \( P(\alpha) \).

Definition
Define the equivalence between AMCs: \( \alpha \sim_{\det} \beta \) if and only if \( P(\alpha) = P(\beta) \).

Question
Does \( \alpha \sim_{\det} \beta \) imply \( \alpha \sim_{\text{O.E.}} \beta \)? Does it imply \( \alpha \sim_{\text{conj}} \beta \)?
The $k$-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.

The graphing $f$ computes a language $L$ in $k$ steps if the $k$-cell decomposition is a refinement of the partition corresponding to $L$.

Show lower bounds by proving a given language is too complex for being refined this way. E.g.:

- Bound the entropy of the graphing and deduce a bound on the number of connected components of the $k$-cell decomposition;
- Produce a given language requiring more connected components.

T. Seiller, CNRS

PRAMs over integers do not compute maxflow efficiently

April 6th, 2019 12/27
Lower Bounds and Entropy

The $k$-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected. The graphing $f$ computes a language $L$ in $k$ steps if the $k$-cell decomposition is a refinement of the partition corresponding to $L$.

Show lower bounds by proving a given language is too complex for being refined this way. E.g.:

- Bound the entropy of the graphing and deduce a bound on the number of connected components of the $k$-cell decomposition;
- Produce a given language requiring more connected components.

T. Seiller, CNRS
The \( k \)-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.

The graphing \( f \) computes a language \( L \) in \( k \) steps if the \( k \)-cell decomposition is a refinement of the partition corresponding to \( L \).

Show lower bounds by proving a given language is too complex for being refined this way. E.g.:

1. Bound the entropy of the graphing and deduce a bound on the number of connected components of the \( k \)-cell decomposition;
2. Produce a given language requiring more connected components.
The $k$-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.

The graphing $f$ computes a language $L$ in $k$ steps if the $k$-cell decomposition is a refinement of the partition corresponding to $L$.

Show lower bounds by proving a given language is too complex for being refined this way. E.g.:

- Bound the entropy of the graphing and deduce a bound on the number of connected components of the $k$-cell decomposition;
- Produce a given language requiring more connected components.

T. Seiller, CNRS

PRAMs over integers do not compute maxflow efficiently

April 6th, 2019 12/27
The $k$-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.

The graphing function $f$ computes a language $L$ in $k$ steps if the $k$-cell decomposition is a refinement of the partition corresponding to $L$.

Show lower bounds by proving a given language is too complex for being refined this way. E.g.:

- Bound the entropy of the graphing and deduce a bound on the number of connected components of the $k$-cell decomposition.
- Produce a given language requiring more connected components.
The $k$-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.

The graphing $f$ computes a language $L$ in $k$ steps if the $k$-cell decomposition is a refinement of the partition corresponding to $L$.

Show lower bounds by proving a given language is too complex for being refined this way. E.g.:

- Bound the entropy of the graphing and deduce a bound on the number of connected components of the $k$-cell decomposition;
- Produce a given language requiring more connected components.
The $k$-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.

The graphing $f$ computes a language $L$ in $k$ steps if the $k$-cell decomposition is a refinement of the partition corresponding to $L$.

Show lower bounds by proving a given language is too complex for being refined this way. E.g.:

- Bound the entropy of the graphing and deduce a bound on the number of connected components of the $k$-cell decomposition;
- Produce a given language requiring more connected components.
The $k$-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.

The graphing $f$ computes a language $L$ in $k$ steps if the $k$-cell decomposition is a refinement of the partition corresponding to $L$.

Show lower bounds by proving a given language is too complex for being refined this way. E.g.:

1. Bound the entropy of the graphing and deduce a bound on the number of connected components of the $k$-cell decomposition;
2. Produce a given language requiring more connected components.
The $k$-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.

The graphing $f$ computes a language $L$ in $k$ steps if the $k$-cell decomposition is a refinement of the partition corresponding to $L$.

Show lower bounds by proving a given language is too complex for being refined this way. E.g.:

1. Bound the entropy of the graphing and deduce a bound on the number of connected components of the $k$-cell decomposition;
2. Produce a given language requiring more connected components.

T. Seiller, CNRS

PRAMs over integers do not compute maxflow efficiently

April 6th, 2019
The $k$-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.
The $k$-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.

The graphing $f$ computes a language $\mathcal{L}$ in $k$ steps if the $k$-cell decomposition is a refinement of the partition corresponding to $\mathcal{L}$. 

Show lower bounds by proving a given language is too complex for being refined this way. E.g.:

1. Bound the entropy of the graphing and deduce a bound on the number of connected components of the $k$-cell decomposition;
2. Produce a given language requiring more connected components.
The $k$-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.

The graphing $f$ computes a language $\mathcal{L}$ in $k$ steps if the $k$-cell decomposition is a refinement of the partition corresponding to $\mathcal{L}$. 

Show lower bounds by proving a given language is too complex for being refined this way. E.g.:

- Bound the entropy of the graphing and deduce a bound on the number of connected components of the $k$-cell decomposition;
- Produce a given language requiring more connected components.
The $k$-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.

The graphing $f$ computes a language $L$ in $k$ steps if the $k$-cell decomposition is a refinement of the partition corresponding to $L$.

Show lower bounds by proving a given language is too complex for being refined this way. E.g.:
The $k$-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.

The graphing $f$ computes a language $\mathcal{L}$ in $k$ steps if the $k$-cell decomposition is a refinement of the partition corresponding to $\mathcal{L}$.

Show lower bounds by proving a given language is too complex for being refined this way. E.g.:

- Bound the entropy of the graphing and deduce a bound on the number of connected components of the $k$-cell decomposition;
The $k$-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.

The graphing $f$ computes a language $\mathcal{L}$ in $k$ steps if the $k$-cell decomposition is a refinement of the partition corresponding to $\mathcal{L}$.

Show lower bounds by proving a given language is too complex for being refined this way. E.g.:

- Bound the entropy of the graphing and deduce a bound on the number of connected components of the $k$-cell decomposition;
- Produce a given language requiring more connected components.
**Lemma**

Let $G$ be a regular deterministic graphing interpreting a PRAM with $p$ processors on $\mathbb{R}^m$. The $k$-th cell decomposition of $X$ w.r.t. $G$ is determined by at most $2^k p^k$ algebraic varieties of degree at most $2^k$. Therefore, by the Milnor-Thom theorem, the $k$-th cell decomposition has at most $(1/2)(2 + 2^{2k} p^k)(1 + 2^{2k} p^k)^{m-1}$ connected components.

... complex arguments showing that maxflow requires more than reasonable connected components ...

**Theorem (Mulmuley 1999)**

Let $G$ be a PRAM$^{\text{w/o}}$ with $2^{O((\log N)^c)}$ processors, where $N$ is the length of the inputs and $c$ any positive integer. Then $G$ does not decide maxflow in $O((\log N)^c)$ steps.
Cucker’s Proof

Lemma

Let $G$ be a regular deterministic graphing interpreting an algebraic circuit of width $p$. The $k$-th cell decomposition of $X$ w.r.t. $G$ is determined by at most $2^k p^k$ algebraic varieties of degree bounded by $2^k$. Thus the number of connected components of the $k$-cell decomposition is bounded by $d(2d−1)^{k+s−1}$.

... exhibiting the right problem ...

Theorem (Cucker 1992)

$\text{NC}_R \neq \text{PTIME}_R$
Another older encounter

Lemma

Let $G$ be a regular deterministic graphing interpreting an algebraic decision tree of max degree $d$. The $k$-th cell decomposition of $X$ w.r.t. $G$ is determined by at most $2^k$ algebraic varieties of degrees bounded by $d$. Thus the number of connected components of the $k$-cell decomposition is bounded by $d(2d-1)^{k+s-1}$.

... provides lower bounds for deciding specific sets ...

Theorem

Set disjointness, i.e. given $A = \{x_1, \ldots, x_n\}$ and $B = \{y_1, \ldots, y_n\}$ deciding whether $A \cap B = \emptyset$ requires at least a depth $\Omega(n \log n)$. 
The Proof
The refined method of Ben-Or

Definition
An algebraic computational tree is defined from nodes $\times$, $+$, $-$, $/$, $\sqrt{}$ and a test node with three sons corresponding to $< 0$, $= 0$ and $> 0$ as in the algebraic decision trees case.

Remark
Without $/$ and $\sqrt{}$ nodes, the previous proof by Steele and Yao provides lower bounds. Ben-Or extension therefore corresponds to dealing with these two operations.
Ben-Or refinement an example

\[
\begin{align*}
  f_1 &= x_1 - x_2 \\
  f_2 &= x_1 - x_3 \\
  f_3 &= x_2 - x_3 \\
  f_4 &= f_1 \times f_2 \\
  f_5 &= f_4 \times f_3 \\
\end{align*}
\]

\[f_5 = 0\]

true \quad false

T. Seiller, CNRS
PRAMs over integers do not compute maxflow efficiently
April 6th, 2019
Ben-Or refinement on an example

\[
\begin{align*}
\text{start} & \\
\quad f_1 &= x_1 - x_2 \\
\quad f_2 &= x_1 - x_3 \\
\quad f_3 &= x_2 - x_3 \\
\quad f_4 &= f_1 \times f_2 \\
\quad f_5 &= f_4 \times f_3 \\
\quad f_5 &= 0
\end{align*}
\]

true \quad false

false
Ben-Or refinement on an example

\[ f_1 = x_1 - x_2 \]
\[ f_2 = x_1 - x_3 \]
\[ f_3 = x_2 - x_3 \]
\[ f_4 = f_1 \times f_2 \]
\[ f_5 = f_4 \times f_3 \]
\[ f_5 = 0 \]
\[ f_5 = 0 \]

If \( f_5 \neq 0 \), then \( f_4 \times f_3 = 0 \).
Ben-Or refinement on an example

\[ f_1 = x_1 - x_2 \]
\[(x_1 - x_2) \times (x_1 - x_3) \times (x_2 - x_3) = 0\]

\[ f_2 = x_1 - x_3 \]
\[ f_1 \times (x_1 - x_3) \times (x_2 - x_3) = 0\]

\[ f_3 = x_2 - x_3 \]
\[ f_1 \times f_2 \times (x_2 - x_3) = 0\]

\[ f_4 = f_1 \times f_2 \]
\[ f_1 \times f_2 \times f_3 = 0\]

\[ f_5 = f_4 \times f_3 \]
\[ f_4 \times f_3 = 0\]

\[ f_5 = 0\]

false

true
Ben-Or refinement on an example

\begin{align*}
  f_1 &= x_1 - x_2 \\
  f_2 &= x_1 - x_3 \\
  f_3 &= x_2 - x_3 \\
  f_4 &= \frac{f_1}{x_1 - x_3} \\
  f_5 &= f_4 \times f_3
\end{align*}

\begin{align*}
  f_1 &= \frac{x_1 - x_2}{x_1 - x_3} \times (x_2 - x_3) = 0 \\
  f_2 &= \frac{f_1}{x_1 - x_3} \times (x_2 - x_3) = 0 \\
  f_3 &= (f_1/f_2) \times (x_2 - x_3) = 0 \\
  f_4 \times f_3 &= 0 \\
  f_5 \times f_3 &= 0
\end{align*}

\begin{align*}
  f_5 &= 0
\end{align*}

true \quad false
Ben-Or refinement on an example

\[ f_1 = x_1 - x_2 \]
\[ f_2 = x_1 - x_3 \]
\[ f_3 = x_2 - x_3 \]
\[ f_4 = f_1 / f_2 \]
\[ f_5 = f_4 \times f_3 \]

\[ f_5 = 0 \]

true \quad false
The proof on graphings: entropic co-trees
The proof on graphings: entropic co-trees
The proof on graphings: entropic co-trees

PRAMs over integers do not compute maxflow efficiently
The proof on graphings: entropic co-trees

\[ e_1^{-1}(\text{yes}) \]

\[ e_1^{-1}(\text{no}) \]

\[ e_2^{-1}(\text{no}) \]

\[ e_2^{-1}(\text{yes}) \]
The proof on graphings: entropic co-trees

\[ e_1 \]

\[ e_2 \]

yes

no
The proof on graphings: entropic co-trees
The proof on graphings: entropic co-trees
The proof on graphings: entropic co-trees

**Theorem**

Let $G$ be a graphing on the AMC of ACTs, $\text{Seq}_k(E)$ the set of length $k$ sequences of edges in $G$, and $D$ its algebraic degree. Suppose $G$ computes the membership problem for $W \subseteq \mathbb{R}^n$ in $k$ steps. Then $W$ has at most

$$\text{Card}(\text{Seq}_k(E)).2^{h_0([G])+1}3^{2kD+n+1}$$

connected components.

**Corollary (Ben-Or 83)**

Let $W \subseteq \mathbb{R}^n$ be any set, and let $N$ be the maximum of the number of connected components of $W$ and $\mathbb{R}^n \setminus W$. An algebraic computation tree computing the membership problem for $W$ has height $\Omega(\log N)$. 
The proof on PRAMs

Two main differences:

- the AMC of PRAMs acts on $\mathbb{Z}^\omega \times (\mathbb{Z}^\omega)^{(\omega)}$, while we acted on $\mathbb{R}^d$ before;
- euclidian division does not translate straightforwardly.

For the first point, we associate to each graphing corresponding to a $\mathbb{Z}$-PRAM a graphing acting on the reals – its real mate – that decides the same integer set. Euclidean division can then be performed on the entropic co-tree by adding variables and relations between them.

Theorem

Let $T$ be a PRAM with $p$ processors, that ends in $k$ steps. The real mate of its $k$-th entropic co-tree is a treeing $Q$ such that:

- $Q$ is of height at most $4k$;
- $\forall x \in \mathbb{Z}^d$, $Q$ accepts $x$ if and only if $T$ accepts $x$;
- the subspace accepted by this treeing can be defined by a set of $(2p)^{4k}$ polynomial equations of degree at most $2^{4k}$.
The proof on PRAMs

**Theorem**

Let $T$ be a PRAM with $p$ processors, that ends in $k$ steps. The real mate of its $k$-th entropic co-tree is a treeing $Q$ such that:

- $Q$ is of height at most $4k$;
- $\forall x \in \mathbb{Z}^d$, $Q$ accepts $x$ if and only if $T$ accepts $x$;
- the subspace accepted by this treeing can be defined by a set of $(2p)^{4k}$ polynomial equations of degree at most $2^{4k}$.

... complex arguments (same as Mulmuley’s proof) showing that maxflow requires more than reasonable connected components ...

**Theorem (Pellissier & Seiller)**

Let $G$ be a $\mathbb{Z}$-PRAM with $2^{O((\log N)^c)}$ processors, where $N$ is the length of the inputs and $c$ any positive integer.

Then $G$ does not decide maxflow in $O((\log N)^c)$ steps.