Time, Space and Monotone Circuits

Christopher Beck

Sept 29 2014

Time and Space

- The two most fundamental computational resources are time and memory (space)

Time and Space

- The two most fundamental computational resources are time and memory (space)
- Many fundamental results in complexity theory concern the relationship between these two

Time and Space

- The two most fundamental computational resources are time and memory (space)
- Many fundamental results in complexity theory concern the relationship between these two
- For even very simple problems, it is sometimes possible to dramatically reduce the space without increasing time much. Other times this doesn't appear to be the case.

Time and Space

- Example: "Element Distinctness". Given a list of n integers (say, $\log^2 n$ bits each), are there any duplicates?

Time and Space

- Example: "Element Distinctness". Given a list of n integers (say, $\log^2 n$ bits each), are there any duplicates?
  - One approach: Sort them ($O(n \log n)$ time, $O(n)$ space), then scan the sorted list for duplicates.

Time and Space

- Example: "Element Distinctness". Given a list of n integers (say, $\log^2 n$ bits each), are there any duplicates?
  - One approach: Sort them ($O(n \log n)$ time, $O(n)$ space), then scan the sorted list for duplicates.
  - Another: Check all $n(n-1)/2$ pairs for equality. ($O(n^2)$ time, $O(\log n)$ space)

Time and Space

- Example: "Element Distinctness". Given a list of n integers (say, $\log^2 n$ bits each), are there any duplicates?
  - One approach: Sort them ($O(n \log n)$ time, $O(n)$ space), then scan the sorted list for duplicates.
  - Another: Check all $n(n-1)/2$ pairs for equality. ($O(n^2)$ time, $O(\log n)$ space)
- Example: "CNF-SAT". Given boolean formula on n variables in CNF, is it satisfiable?

Time and Space

- Example: "Element Distinctness". Given a list of n integers (say, $\log^2 n$ bits each), are there any duplicates?
  - One approach: Sort them ($O(n \log n)$ time, $O(n)$ space), then scan the sorted list for duplicates.
  - Another: Check all $n(n-1)/2$ pairs for equality. ($O(n^2)$ time, $O(\log n)$ space)
- Example: "CNF-SAT". Given boolean formula on n variables in CNF, is it satisfiable?
  - Trivial algorithm: Exhaustively enumerate all assignments and check. ($O(2^n)$ time, $O(n)$ space)

Time and Space

- Example: "Element Distinctness". Given a list of n integers (say, $\log^2 n$ bits each), are there any duplicates?
  - One approach: Sort them ($O(n \log n)$ time, $O(n)$ space), then scan the sorted list for duplicates.
  - Another: Check all $n(n-1)/2$ pairs for equality. ($O(n^2)$ time, $O(\log n)$ space)
- Example: "CNF-SAT". Given boolean formula on n variables in CNF, is it satisfiable?
  - Trivial algorithm: Exhaustively enumerate all assignments and check. ($O(2^n)$ time, $O(n)$ space)
  - If ETH holds, then there is no $O(2^{(1-\epsilon)n})$ time algorithm, even with exponential space.

Time and Space

- Basic question: Which problems require large amounts of memory to be solved efficiently?

Time and Space

- Basic question: Which problems require large amounts of memory to be solved efficiently?
- Which problems have true time-space trade-offs? Can we explain why they do or don't?

Time and Space

- Basic question: Which problems require large amounts of memory to be solved efficiently?
- Which problems have true time-space trade-offs? Can we explain why they do or don't?
- Besides concrete problems, the important meta-algorithm "Dynamic Programming" always trades space for time.

Time and Space

- Basic question: Which problems require large amounts of memory to be solved efficiently?
- Which problems have true time-space trade-offs? Can we explain why they do or don't?
- Besides concrete problems, the important meta-algorithm "Dynamic Programming" always trades space for time.
- Basic question: What are the limits of the strategy?

Known Tradeoffs in Concrete Models

- Questions like these have been asked almost since the inception of the field.
- Much work in “decision tree” / “branching program” models.

Known Tradeoffs in Concrete Models

- Questions like these have been asked almost since the inception of the field.
- Much work in “decision tree” / “branching program” models.
  - Sorting (in a “structured” model) [Borodin, Cook ’82]

Known Tradeoffs in Concrete Models

- Questions like these have been asked almost since the inception of the field.
- Much work in “decision tree” / “branching program” models.
  - Sorting (in a “structured” model) [Borodin, Cook ’82]
  - Element distinctness [Beame ’91]

Known Tradeoffs in Concrete Models

- Questions like these have been asked almost since the inception of the field.
- Much work in “decision tree” / “branching program” models.
  - Sorting (in a “structured” model) [Borodin, Cook ’82]
  - Element distinctness [Beame ’91]
    - (Our earlier examples are tight, in “R-way branching programs”)

Known Tradeoffs in Concrete Models

- Questions like these have been asked almost since the inception of the field.
- Much work in "decision tree" / "branching program" models.
  - Sorting (in a "structured" model) [Borodin, Cook '82]
  - Element distinctness [Beame '91]
    - (Our earlier examples are tight, in "R-way branching programs")
  - Explicit function in P which only has subexponential size boolean branching programs with superlinear length [Ajtai '99].

Known Tradeoffs in Concrete Models

- Questions like these have been asked almost since the inception of the field.
- Much work in "decision tree" / "branching program" models.
  - Sorting (in a "structured" model) [Borodin, Cook '82]
  - Element distinctness [Beame '91]
    - (Our earlier examples are tight, in "R-way branching programs")
  - Explicit function in P which only has subexponential size boolean branching programs with superlinear length [Ajtai '99].
  - Extension to inapproximiability for randomized branching programs. [Beame, Saks, Sun, Vee '02].

Known Tradeoffs in Concrete Models

- Questions like these have been asked almost since the inception of the field.
- Much work in "decision tree" / "branching program" models.
- Much work on "pebble games" on DAGs
  - Given a directed acyclic graph, a "pebble" may be placed on any node, all of whose predecessors are pebbled, and removed at any time. The goal is to pebble all nodes, in some order, using few pebbles.

Known Tradeoffs in Concrete Models

- Questions like these have been asked almost since the inception of the field.
- Much work in “decision tree” / “branching program” models.
- Much work on “pebble games” on DAGs
  - Given a directed acyclic graph, a “pebble” may be placed on any node, all of whose predecessors are pebbled, and removed at any time. The goal is to pebble all nodes, in some order, using few pebbles.
  - Pebble games model the process of allocating registers to compute the result of a specific circuit.

Known Tradeoffs in Concrete Models

- Questions like these have been asked almost since the inception of the field.
- Much work in "decision tree" / "branching program" models.
- Much work on "pebble games" on DAGs
  - Given a directed acyclic graph, a "pebble" may be placed on any node, all of whose predecessors are pebbled, and removed at any time. The goal is to pebble all nodes, in some order, using few pebbles.
  - Pebble games model the process of allocating registers to compute the result of a specific circuit.
  - Theme: High connectivity inhibits efficient small space algorithms.

Known Tradeoffs in Concrete Models

- Questions like these have been asked almost since the inception of the field.
- Much work in “decision tree” / “branching program” models.
- Much work on “pebble games” on DAGs
  - Given a directed acyclic graph, a “pebble” may be placed on any node, all of whose predecessors are pebbled, and removed at any time. The goal is to pebble all nodes, in some order, using few pebbles.
  - Pebble games model the process of allocating registers to compute the result of a specific circuit.
  - Theme: High connectivity inhibits efficient small space algorithms.
    - (Paul, Tarjan ’77) DAGs built from expanders become exponentially difficult to pebble with reduced space.

Known Tradeoffs in Concrete Models

- Questions like these have been asked almost since the inception of the field.
- Much work in "decision tree" / "branching program" models.
- Much work on "pebble games" on DAGs
  - Given a directed acyclic graph, a "pebble" may be placed on any node, all of whose predecessors are pebbled, and removed at any time. The goal is to pebble all nodes, in some order, using few pebbles.
  - Pebble games model the process of allocating registers to compute the result of a specific circuit.
  - Theme: High connectivity inhibits efficient small space algorithms.
    - ⋆ (Paul, Tarjan '77) DAGs built from expanders become exponentially difficult to pebble with reduced space.
    - ⋆ (Lipton, Tarjan '80) Any planar DAG of degree $O(1)$ with n vertices can be pebbled with $n^{2/3}$ pebbles, and $n^{5/3}$ steps.

Known Tradeoffs in Concrete Models

- Questions like these have been asked almost since the inception of the field.
- Much work in "decision tree" / "branching program" models.
- Much work on "pebble games" on DAGs
  - Given a directed acyclic graph, a "pebble" may be placed on any node, all of whose predecessors are pebbled, and removed at any time. The goal is to pebble all nodes, in some order, using few pebbles.
  - Pebble games model the process of allocating registers to compute the result of a specific circuit.
  - Theme: High connectivity inhibits efficient small space algorithms.
    - ⋆ (Paul, Tarjan '77) DAGs built from expanders become exponentially difficult to pebble with reduced space.
    - ⋆ (Lipton, Tarjan '80) Any planar DAG of degree $O(1)$ with n vertices can be pebbled with $n^{2/3}$ pebbles, and $n^{5/3}$ steps.
    - ⋆ (Alon, Seymour, Thomas '90) Any DAG of degree $O(1)$ free of K_h-minors can be pebbled with $h^{3/2}n^{1/2}$ pebbles.

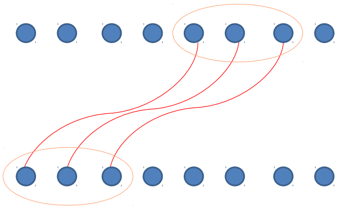
Known Tradeoffs in Concrete Models

- Questions like these have been asked almost since the inception of the field.
- Much work in "decision tree" / "branching program" models.
- Much work on "pebble games" on DAGs
- For (restricted) circuits (AC^0, monotone, algebraic...)

Known Tradeoffs in Concrete Models

- Questions like these have been asked almost since the inception of the field.
- Much work in “decision tree” / “branching program” models.
- Much work on “pebble games” on DAGs
- For (restricted) circuits (AC^0, monotone, algebraic...)
  - Valiant ['76] shows that any algebraic circuit over a finite field computing a linear transformation whose matrix has the property that, any square submatrix is full rank, has the graph-theoretic property of being a 'superconcentrator'.

Superconcentrator



Definition

A *superconcentrator* of capacity n is a DAG $G = (V, E)$ with two disjoint sets of vertices $I, O \subset V$, $|I| = |O| = n$ such that for all subsets $I' \subseteq I, O' \subseteq O$ with $|I'| = |O'|$, there exist $|I'|$ vertex-disjoint paths connecting I' and O'.

Known Tradeoffs in Concrete Models

- Questions like these have been asked almost since the inception of the field.
- Much work in "decision tree" / "branching program" models.
- Much work on "pebble games" on DAGs
- For (restricted) circuits (AC^0, monotone, algebraic...)
  - Valiant ['76] shows that for any algebraic circuit over a finite field (or any) computing a linear transformation, whose matrix has the property that any square submatrix is full rank, has the graph-theoretic property of being a 'superconcentrator'.
  - Savage ['77], Tompa ['81], others use such arguments to show that the Fourier Transform and similar e.g. cannot be computed with space $n^{1-\epsilon}$ without using time $n^{1+\epsilon}$.

Known Tradeoffs in Concrete Models

- In Proof Complexity, no trade-offs known until much more recently.

Known Tradeoffs in Concrete Models

- In Proof Complexity, no trade-offs known until much more recently.
- (Ben-Sasson, Nordstrom '08) First tradeoffs of any kind, in Resolution, for sublinear space.

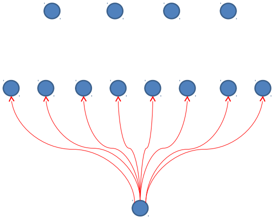
Known Tradeoffs in Concrete Models

- In Proof Complexity, no trade-offs known until much more recently.
- (Ben-Sasson, Nordstrom '08) First tradeoffs of any kind, in Resolution, for sublinear space.
- (B., Beame, Impagliazzo '12) Tradeoffs even up to exponential space, with superpolynomial blowups in time.

Known Tradeoffs in Concrete Models

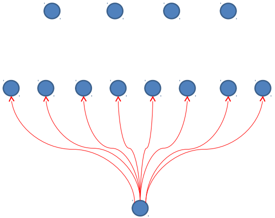
- In Proof Complexity, no trade-offs known until much more recently.
- (Ben-Sasson, Nordstrom '08) First tradeoffs of any kind, in Resolution, for sublinear space.
- (B., Beame, Impagliazzo '12) Tradeoffs even up to exponential space, with superpolynomial blowups in time.
- This result is different from previous time-space tradeoff results in that it technically extends the previously known (tight) lower bounds for time. It is a purely combinatorial argument and doesn't reduce to pebbling.

Bottleneck Counting Argument



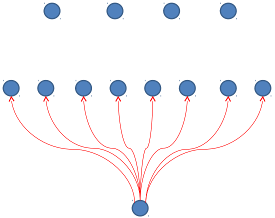
- Many lower bounds follow Haken ['85]'s *bottleneck counting* scheme.

Bottleneck Counting Argument



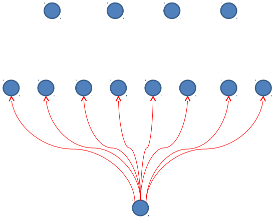
- Many lower bounds follow Haken ['85]'s *bottleneck counting* scheme.
  - Map *input assignments* to *gates* of the circuit and prove that the probability that any assignment goes to a particular gate is small, e.g. $< \epsilon$.

Bottleneck Counting Argument



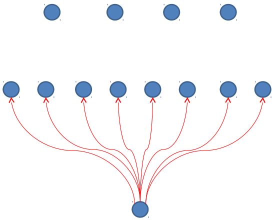
- Many lower bounds follow Haken ['85]'s *bottleneck counting* scheme.
  - Map *input assignments* to *gates* of the circuit and prove that the probability that any assignment goes to a particular gate is small, e.g. $< \epsilon$.
  - Conclude that there are at least ϵ^{-1} gates.

Bottleneck Counting Argument



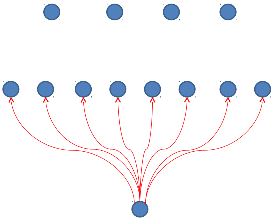
- In Haken's work this map is ad hoc.

Bottleneck Counting Argument



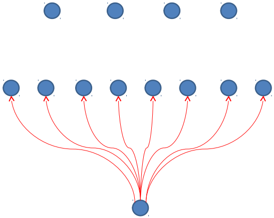
- In Haken's work this map is ad hoc.
- In the "method of random restrictions", it's based on the way in which the circuit simplifies when many inputs are fixed.

Bottleneck Counting Argument



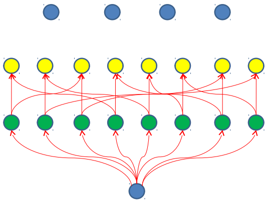
- In Haken's work this map is ad hoc.
- In the "method of random restrictions", it's based on the way in which the circuit simplifies when many inputs are fixed.
- In Razborov's "method of approximations", it's based on successive approximations (of low complexity) to the gates of the circuit.

Bottleneck Counting Argument



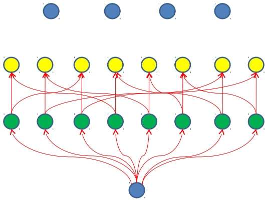
- In Haken's work this map is ad hoc.
- In the "method of random restrictions", it's based on the way in which the circuit simplifies when many inputs are fixed.
- In Razborov's "method of approximations", it's based on successive approximations (of low complexity) to the gates of the circuit.
- Janos Simon ['97, et.al '13] points out the commonalities of these.

Bottlenecks and Tradeoffs



In [BBI'12], we derived a time space tradeoff, starting from a tight running time lower bound, which we might sketch as follows:

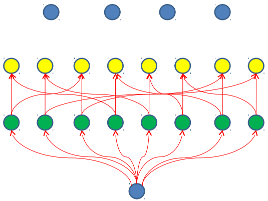
Bottlenecks and Tradeoffs



In [BBI'12], we derived a time space tradeoff, starting from a tight running time lower bound, which we might sketch as follows:

- Consider a short proof, and consider the map from the size lower bound. By varying the parameters to define it, obtain a second map. We have $\Pr_{\vec{x}}[f_1(\vec{x}) = g] \leq \epsilon$ and $\Pr_{\vec{x}}[f_2(\vec{x}) = g] \leq \epsilon$ for all gates g.

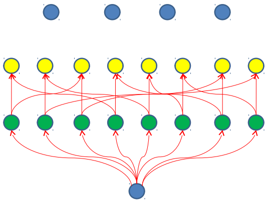
Bottlenecks and Tradeoffs



In [BBI'12], we derived a time space tradeoff, starting from a tight running time lower bound, which we might sketch as follows:

- Consider a short proof, and consider the map from the size lower bound. By varying the parameters to define it, obtain a second map. We have $\Pr_{\vec{x}}[f_1(\vec{x}) = g] \leq \epsilon$ and $\Pr_{\vec{x}}[f_2(\vec{x}) = g] \leq \epsilon$ for all gates g.
- However, now prove also that for any g_1, g_2 that $\Pr_{\vec{x}}[f_1(\vec{x}) = g_1 \wedge f_2(\vec{x}) = g_2] \leq \epsilon^2$.

Bottlenecks and Tradeoffs



In [BBI'12], we derived a time space tradeoff, starting from a tight running time lower bound, which we might sketch as follows:

- Consider a short proof, and consider the map from the size lower bound. By varying the parameters to define it, obtain a second map. We have $\Pr_{\vec{x}}[f_1(\vec{x}) = g] \leq \epsilon$ and $\Pr_{\vec{x}}[f_2(\vec{x}) = g] \leq \epsilon$ for all gates g.
- However, now prove also that for any g_1, g_2 that $\Pr_{\vec{x}}[f_1(\vec{x}) = g_1 \wedge f_2(\vec{x}) = g_2] \leq \epsilon^2$.
- If the size of the DAG is $\approx \epsilon^{-1}$, we have a weak form of expansion, and morally it implies a space lower bound.

Open Questions

- Can we use arguments like this to extend monotone circuit size lower bounds to time-space tradeoffs?

Open Questions

- Can we use arguments like this to extend monotone circuit size lower bounds to time-space tradeoffs?
- What about in algebraic circuits?

Open Questions

- Can we use arguments like this to extend monotone circuit size lower bounds to time-space tradeoffs?
- What about in algebraic circuits?

Thanks!