Las Vegas Algorithms for Linear (and Integer) Programming when the Dimension is Small

Kenneth L. Clarkson
presented by Susan Martonosi

September 29, 2003

Outline

- Applications of the algorithm
- Previous work
- Assumptions and notation
- Algorithm 1: “Recurrent Algorithm”
- Algorithm 2: “Iterative Algorithm”
- Algorithm 3: “Mixed Algorithm”
- Contribution of this paper to the field
Applications of the Algorithms

Algorithms give a bound that is “good” in $n$ (number of constraints), but “bad” in $d$ (dimension). So we require the problem to have a small dimension.

- **Chebyshev approximation:** fitting a function by a rational function where both the numerator and denominator have relatively small degree. The dimension is the sum of the degrees of the numerator and denominator.
- **Linear separability:** separating two sets of points in $d$-dimensional space by a hyperplane
- **Smallest enclosing circle problem:** find a circle of smallest radius that encloses points in $d$ dimensional space
**Previous work**

- **Megiddo**: Deterministic algorithm for LP in $O(2^d n)$
- **Clarkson; Dyer**: $O(3^{d^2}n)$
- **Dyer and Frieze**: Randomized algo. with expected time no better than $O(d^{3d} n)$
- **This paper’s “mixed” algo.**: Expected time
  \[ O(d^2 n) + (\log n)O(d)^{d/2+O(1)} + O(d^4 \sqrt{n} \log n) \]  
as $n \to \infty$
Assumptions

- Minimize $x_1$ subject to $Ax \leq b$
- The polyhedron $\mathcal{F}(A, b)$ is non-empty and bounded and $0 \in \mathcal{F}(A, b)$
- The minimum we seek occurs at a unique point, which is a vertex of $\mathcal{F}(A, b)$
  - If a problem is bounded and has multiple optimal solutions with optimal value $x_1^*$, choose the one with the minimum Euclidean norm
    $$\min\{\|x\|_2 | x \in \mathcal{F}(A, b), x_1 = x_1^*\}$$
- Each vertex of $\mathcal{F}(A, b)$ is defined by $d$ or fewer constraints
Let:

- $H$ denote the set of constraints defined by $A$ and $b$
- $\mathcal{O}(S)$ be the optimal value of the objective function for the LP defined on $S \subseteq H$
- “Each vertex of $\mathcal{F}(A, b)$ is defined by $d$ or fewer constraints” implies that $\exists \mathcal{B}(H) \subset H$ of size $d$ or less such that $\mathcal{O}(\mathcal{B}(H)) = \mathcal{O}(H)$. We call this subset $\mathcal{B}(H)$ the basis of $H$. All other constraints in $H \setminus \mathcal{B}(H)$ are redundant.
- a constraint $h \in H$ be called extreme if $\mathcal{O}(H \setminus h) < \mathcal{O}(H)$ (these are the constraints in $\mathcal{B}(H)$).
Algorithm 1: Recursive

- Try to eliminate redundant constraints
- Once our problem has a small number of constraints \((n \leq 9d^2)\), then use Simplex to solve it.
- Build up a smaller set of constraints that eventually include all of the extreme constraints and a small number of redundant constraints
  - Choose \(r = d\sqrt{n}\) unchosen constraints of \(H \setminus S\) at random
  - Recursively solve the problem on the subset of constraints, \(R \cup S\)
  - Determine which remaining constraints \((V)\) are violated by this optimal solution
  - Add \(V\) to \(S\) if it’s not too big (\(|V| \leq 2\sqrt{n}\)).
  - Otherwise, if \(V\) is too big, then pick \(r\) new constraints

We stop once \(V\) is empty: we’ve found a set \(S \cup R\) such that no other constraints in \(H\) are violated by its optimal solution. This optimal solution \(x\) is thus optimal for the original problem.
Recursive Algorithm

**Input:** A set of constraints \( H \). **Output:** The optimum \( \mathcal{B}(H) \)

1. \( S \leftarrow \emptyset; \ C_d \leftarrow 9d^2 \)
2. If \( n \leq C_d \) return Simplex\((H)\)
2.1 else repeat:
   - choose \( R \subset H \setminus S \) at random, with \( |R| = r = d\sqrt{n} \)
   - \( x \leftarrow \text{Recursive}(R \cup S) \)
   - \( V \leftarrow \{ h \in H \mid \text{vertex defined by } x \text{ violates } h \} \)
   - if \( |V| \leq 2\sqrt{n} \) then \( S \leftarrow S \cup V \)
   - until \( V = \emptyset \)
2.2 return \( x \)
Recursive Algorithm: Proof Roadmap

Questions:

- How do we know that $S$ doesn’t get too large before it has all extreme constraints?
- How do we know we will find a set of violated constraints $V$ that’s not too big (i.e. the loop terminates quickly)?

Roadmap:

**Lemma 1.** If the set $V$ is nonempty, then it contains a constraint of $\mathcal{B}(H)$.

**Lemma 2.** Let $S \subseteq H$ and let $R \subseteq H \setminus S$ be a random subset of size $r$, with $|H \setminus S| = m$. Let $V \subseteq H$ be the set of constraints violated by $O(R \cup S)$. Then the expected size of $V$ is no more than $\frac{d(m-r+1)}{r-d}$.

And we’ll use this to show the following Lemma:
Lemma 3. The probability that any given execution of the loop body is ”successful” ($|V| \leq 2\sqrt{n}$ for this recursive version of the algorithm) is at least $1/2$, and so on average, two executions or less are required to obtain a successful one.

This will leave us with a running time

$$T(n, d) \leq 2dT(3d\sqrt{n}, d) + O(d^2n) \text{ for } n > 9d^2.$$
Recursive Algorithm: Proof of Lemma 1

Proof. Lemma 1: When $V$ is nonempty, it contains a constraint of $\mathcal{B}(H)$.

Suppose on the contrary that $V \neq \emptyset$ contains no constraints of $\mathcal{B}(H)$.

Let a point $x \preceq y$ if $(x_1, \|x\|_2) \xleftarrow{L} (y_1, \|y\|_2)$ ($x$ is better than $y$).

Let $x^*(T)$ be the optimal solution over a set of constraints $T$. Then $x^*(R \cup S)$ satisfies all the constraints of $\mathcal{B}(H)$ (it is feasible), and thus $x^*(R \cup S) \succeq x^*(\mathcal{B}(H))$.

However, since $R \cup S \subset H$, we know that $x^*(R \cup S) \preceq x^*(H) = x^*(\mathcal{B}(H))$. Thus, $x^*(R \cup S)$ has the same obj. fcn value and norm as $x^*(\mathcal{B}(H))$. By the uniqueness of this point, $x^*(R \cup S) = x^*(\mathcal{B}(H)) = x^*(H)$, and $V = \emptyset$. Contradiction!

So, every time $V$ is added to $S$, at least one extreme constraint of $H$ is added (so we’ll do this at most $d$ times).
Recursive Algorithm: Proof of Lemma 2

Proof. Lemma 2: The expected size of $V$ is no more than $\frac{d(m-r+1)}{r-d}$.

First assume problem nondegenerate.

Let $C_H = \{x^*(T \cup S) | T \subseteq H \setminus S\}$, subset of optima.

Let $C_R = \{x^*(T \cup S) | T \subseteq R\}$

The call Recursive($R \cup S$) returns an element $x^*(R \cup S)$:

- an element of $C_H$
- unique element of $C_R$ satisfying every constraint in $R$. 
Recursive Algorithm: Proof of Lemma 2

Choose $x \in C_H$ and let $v_x =$ number of constraints in $H$ violated by $x$.

$$E[|V|] = E[\sum_{x \in C_H} v_x I(x = x^*(R \cup S))] = \sum_{x \in C_H} v_x P_x$$

where

$$I(x = x^*(R \cup S)) = \begin{cases} 1 & \text{if } x = x^*(R \cup S) \\ 0 & \text{otherwise} \end{cases}$$

and $P_x = P(x = x^*(R \cup S))$

How to find $P_x$?
Recursive Algorithm: Proof of Lemma 2

Let $N =$ number of subsets of $H \setminus S$ of size $r$ s.t. $x^*(\text{subset}) = x^*(R \cup S)$.

Then $N = \binom{m}{r} P_x$ and $P_x = \frac{N}{\binom{m}{r}}$.

To find $N$, note that $x^*(\text{subset}) \in \mathcal{C}_H$ and $x^*(\text{subset}) = x^*(R \cup S)$ only if

- $x^*(\text{subset}) \in \mathcal{C}_R$ as well
- $x^*(\text{subset})$ satisfies all constraints of $R$

Therefore, $N =$ No. of subsets of $H \setminus S$ of size $r$ s.t. $x^*(\text{subset}) \in \mathcal{C}_R$ and $x^*(\text{subset})$ satisfies all constraints of $R$. 
Recursive Algorithm: Proof of Lemma 2

For some such subset of $H\setminus S$ of size $r$ and such that $x^*(\text{subset}) = x^*(R \cup S)$, let $T$ be the *minimal* set of constraints such that $x^*(\text{subset}) = x^*(T \cup S)$.

- $x^*(\text{subset}) \in C_R$ implies $T \subseteq R$
- nondegeneracy implies $T$ is unique and $|T| \leq d$

Let $i_x = |T|$.

In order to have $x^*(T \cup S) = x^*(R \cup S)$ (and thus $x^*(\text{subset}) = x^*(R \cup S)$), when constructing our subset we must choose:

- the $i_x$ constraints of $T \subseteq R$
- $r - i_x$ constraints from $H\setminus S\setminus T\setminus V$
Therefore, $N = (\binom{m-vx-ix}{r-ix})$ and $P_x = \frac{(\binom{m-vx-ix}{r-ix})}{\binom{m}{r}} \leq \frac{m-r+1}{r-d} \frac{(\binom{m-vx-ix}{r-ix-1})}{\binom{m}{r}}$

$E[|V|] \leq \frac{m-r+1}{r-d} \sum_{x \in C_H} v_x \frac{(\binom{m-vx-ix}{r-ix-1})}{\binom{m}{r}} \leq d \frac{m-r+1}{r-d}$

(where the summand is $E[\text{No. of } x \in C_R \text{ violating exactly one constraint in } R] \leq d$)

For the degenerate case, we can perturb the vector $b$ by adding $(\epsilon, \epsilon^2, \ldots, \epsilon^n)$ and show that the bound on $|V|$ holds for this perturbed problem, and that the perturbed problem has at least as many violated constraints as the original degenerate problem.

$\square$
Recursive Algorithm: Proof of Lemma 3

Proof. Lemma 3: $P(\text{successful execution}) \geq 1/2$; $E[\text{Executions til 1st success}] \leq 2$.

Here, $P(\text{unsuccessful execution}) = P(|V| > 2\sqrt{n})$

$2E[|V|] \leq 2d^{m-r+1} = 2^{n-d\sqrt{n}+1}$ (since $r = d\sqrt{n}) \leq 2\sqrt{n}$

So, $P(\text{unsuccessful execution}) = P(|V| > 2\sqrt{n}) \leq P(|V| > 2E[|V|]) \leq 1/2$, by the Markov Inequality.

$P(\text{successful execution}) \geq 1/2$, and the expected number of loops until our first successful execution is less than 2. \qed

\text{16}
Recursive Algorithm: Running Time

As long as $n > 9d^2$,

- Have at most $d + 1$ augmentations to $S$ (successful iterations), with expected 2 tries until success
- With each success, $S$ grows by at most $2\sqrt{n}$, since $|V| \leq 2\sqrt{n}$
- After each success, we run the Recursive algorithm on a problem of size $|S \cup R| \leq 2d\sqrt{n} + d\sqrt{n} = 3d\sqrt{n}$
- After each recursive call, we check for violated constraints, which takes $O(nd)$ each of at most $d + 1$ times

$$T(n, d) \leq 2(d + 1)T(3d\sqrt{n}, d) + O(d^2n), \text{ for } n > 9d^2$$
Algorithm 2: Iterative

- Doesn’t call itself, calls Simplex directly each time
- Associates weight $w_h$ to each constraint which determines the probability with which it is selected
- Each time a constraint is violated, its weight is doubled
- Don’t add $V$ to a set $S$; rather reselect $R$ (of size $9d^2$) over and over until it includes the set $\mathcal{B}(H)$
Algorithm 2: Iterative

**Input:** A set of constraints $H$. **Output:** The optimum $\mathcal{B}(H)$

1. $\forall h \in H$, $w_h \leftarrow 1$; $C_d = 9d^2$
2. If $n \leq C_d$, return $\text{Simplex}(H)$
   2.1 else repeat:
      - choose $R \subset H$ at random, with $|R| = r = C_d$
      - $x \leftarrow \text{Simplex}(R)$
      - $V \leftarrow \{h \in H| \text{vertex defined by } x \text{ violates } h\}$
      - if $w(V) \leq 2 \frac{w(H)}{9d-1}$ then for $h \in V$, $w_h \leftarrow 2w_h$
      - until $V = \emptyset$
2.2 return $x$
Iterative Algorithm: Analysis

- Lemma 1: “If the set $V$ is nonempty, then it contains a constraint of $\mathcal{B}(H)$” still holds (proof as above with $S = \emptyset$).

- Lemma 2: “Let $S \subseteq H$ and let $R \subseteq H \setminus S$ be a random subset of size $r$, with $|H \setminus S| = m$. Let $V \subseteq H$ be the set of constraints violated by $\mathcal{O}(R \cup S)$. Then the expected size of $V$ is no more than $\frac{d(m-r+1)}{r-d}$,” still holds with the following changes. Consider each weight-doubling as the creation of multinodes. So “size” of a set is actually its weight. So we have $S = \emptyset$, and thus $|H \setminus S| = m = w(H)$. This gives us $E[w(V)] \leq \frac{d(w(H)-9d^2+1)}{9d^2-d} \leq \frac{w(H)}{9d-1}$

- Lemma 3: If we define a “successful iteration” to be $w(V) \leq 2\frac{w(H)}{9d-1}$, then Lemma 3 holds, and the probability that any given execution of the loop body is ”successful” is at least $1/2$, and so on average, two executions or less are required to obtain a successful one.
Iterative Algorithm: Running Time

The Iterative Algorithm runs in $O(d^2 n \log n) + (d \log n)O(d^{d/2+O(1)})$ expected time, as $n \to \infty$, where the constant factors do not depend on $d$.

First start by showing expected number of loop iterations $= O(d \log n)$

- By Lemma 3.1, at least one extreme constraint $h \in \mathcal{B}(H)$ is doubled during a successful iteration
- Let $d' = |\mathcal{B}(H)|$. After $kd'$ successful executions $w(\mathcal{B}(H)) = \sum_{h \in \mathcal{B}(H)} 2^{n_h}$, where $n_h$ is the number of times $h$ entered $V$ and thus $\sum_{h \in \mathcal{B}(H)} n_h \geq kd'$
- $\sum_{h \in \mathcal{B}(H)} w_h \geq \sum_{h \in \mathcal{B}(H)} 2^k = d'2^k$
- When members of $V$ are doubled, increase in $w(H) = w(V) \leq \frac{2}{9d-1}$, so after $kd'$ successful iterations, we have

$w(H) \leq n(1 + \frac{2}{9d-1})^{kd'} \leq ne^{\frac{2kd'}{9d-1}}$
• $V$ sure to be empty when $w(B(H)) > w(H)$ (i.e. $P(\text{Choose } B(H)) > 1$). This gives us:

$$k > \frac{\ln(n/d')}{\ln 2 - \frac{2d}{9d-1}}, \text{ or } kd' = O(d \log n) \text{ successful iterations} = O(d \log n) \text{ iterations.}$$

Within a loop:

• Can select a sample $R$ in $O(n)$ time [Vitter '84]
• Determining violated constraints, $V$, is $O(dn)$
• Simplex algorithm takes $d^{O(1)}$ time per vertex, times $\binom{2Cd}{\lfloor d/2 \rfloor}$ vertices [?]. Using Stirling’s approximation, this gives us $O(d)^{d/2+O(1)}$ for Simplex

Total running time:

$$O(d \log n) \times [O(dn) + O(d)^{d/2+O(1)}] = O(d^2 n \log n) + (d \log n)O(d)^{d/2+O(1)}$$
Algorithm 3: Mixed

- Follow the Recursive Algorithm, but rather than calling itself, call the Iterative Algorithm instead.
- Runtime of Recursive: $T(n, d) \leq 2(d + 1)T(3d\sqrt{n}, d) + O(d^2n)$, for $n > 9d^2$
- In place of $T(3d\sqrt{n})$, substitute in runtime of Iterative algorithm on $3d\sqrt{n}$ constraints.
- Runtime of Mixed Algorithm: $O(d^2n) + (d^2 \log n)O(d)^{d/2+O(1)} + O(d^4\sqrt{n} \log n)$
Contributions of this paper to the field

- Leading term in dependence on $n$ is $O(d^2 n)$, an improvement over $O(d^{3d} n)$
- Algorithm can also be applied to integer programming (Jan’s talk)
- Algorithm was later applied as overlying algorithm to “incremental” algorithms (Jan’s talk) to give a sub-exponential bound for linear programming (rather than using Simplex once $n \leq 9d^2$, use an incremental algorithm)