I. Integer programming part of Clarkson-paper

II. Incremental Linear Programming, Section 9.10.1 in Randomized Algorithms-book

presented by Jan De Mot

September 29, 2003

This presentation is based on: Clarkson, Kenneth L. *Las Vegas Algorithms for Linear and Integer Programming When the Dimension is Small. Journal of the ACM* 42(2), March 1995, pp. 488-499. Preliminary version in Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science, 1988.

and Chapter 9 of: Motwani, Rajeev, and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

Outline

Part I: Integer Linear Programming (ILP)

- Previous work
- Algorithm for solving Integer Linear Programs [Clarkson 1995] based on the mixed algorithm for LP (Susan)
  - Concept
  - Running Time Analysis

Part II: Incremental Linear Programming

- Concept
- SeideLP [Seidel 1991]
- BasisLP [Sharir and Welzl 1992]

Part I: Integer Linear Programming

Previous Work

- [Lenstra 1983] showed how to solve an ILP in polynomial time when the numbers of variables is fixed.
- Subsequent improvements (e.g. by [Frank and Tardos 1987]) show that the fasted deterministic algorithm requires $d^{O(d)}n\phi$ operations on $d^{O(1)}\phi$ -bit numbers.

- Running time of *new* ILP algorithm: $O(2^d + 8^d\sqrt{n \ln n} \ln n)$
  This is substantially faster than Lenstra's for $n \gg d$.

ILP Problem

- Find the optimum of:

$$\max\{\mathbf{cx} | \mathbf{Ax} \leq \mathbf{b}; \mathbf{x} \text{ integral}\},$$

where $\mathbf{A} \in Q^{n \times d}$, $\mathbf{b} \in Q^n$ and $\mathbf{c} \in Q^d$.

Notation and Preliminaries

- Let:
  - H denote the set of constraints defined by $\mathbf{A}$ and b,
  - $\mathbf{x}^*(S)$ denote the optimal solution of the ILP defined on $S \subseteq H$ (not the corresponding LP relaxation).
- Assume:
  - *Bounded solution* by adding to H a new set of $2d$ constraints $\widehat{H}$:

    $$|x_i| \leq 2^{K_d} + 1, \text{ for } 1 \leq i \leq d,$$

    where $K_d = 2d^2\phi + \lceil log_2(n+1) \rceil$, and where we use a result by [Schrijver 1986]: if an ILP has finite solution, then every coordinate of that optimum has size no more than K_d where ϕ is the facet complexity of $\mathcal{F}(\mathbf{A}, \mathbf{b})$.
  - *Unique solution* by choosing the lexicographically largest point achieving the optimum value.

ILP Algorithm: Concept

- First it is established that an optimum is determined by a *small* set ([Bell 1977] and [Scarf 1977]):

  Lemma: *There is a set* $H^* \subseteq H$ *with* $|H^*| \leq 2^d - 1$ *and with* $\mathbf{x}^*(H) = \mathbf{x}^*(H^*)$

- ILP algorithms are variations on the LP algorithms, with sample sizes using 2^d rather than d and using Lenstra's algorithm in the base case.
- Here, we convert the mixed algorithm for LPs to a mixed algorithm for ILPs, establishing the right sample sizes and criteria for successful iterations in both the recursive and iterative part of the mixed algorithm.

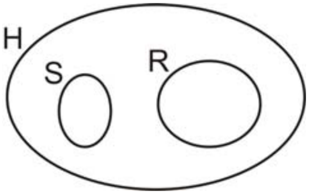
ILP Algorithm: Details

- Lemma 2, related to the LP recursive algorithm, needs to be redone due to the fact that H^* is not unique.
- Reminder: why do we need lemma 2?

  We want to make sure the set of violated constraints V does not become too big.
- **Lemma 2** (ILP version): *Let* $S \subset H$, *and let* $R \subset H \setminus S$ *be a random subset of size* $r > 2^{d+1}$, *with* $|H \setminus S| = n$. *Let* $V \subset S$ *be the set of constraints violated by* $\mathbf{x}^*(R \cup S)$. *Then with probability* $1/2$, $|V| \leq 2^{d+1} n (\ln r)/r$.
- Other necessary lemma's remain valid or can be adapted easily, yielding the following essential parameters for the ILP mixed algorithm:
  - Recursive part: $r = 2^d \sqrt{2n \ln n}$, use Lenstra's algorithm for $n \leq 2^{2d+5} d$, and require $|V| \leq \sqrt{2n \ln n}$ for a successful iteration.
  - Iterative part: $r = 2^{2d+4}(2d+4)$, with a corresponding $|V|$ bound of $n(\ln 2)/2^{d+3}$.

ILP Algorithm: Proof of Lemma 2 (ILP version)

- *Proof*. Lemma 2 (ILP version): With probability $1/2$, $|V| \leq 2^{d+1} n (\ln r)/r$.
- Assume S is empty. For S not empty: similar proof. Let $m = 2^d - 1$, and let v_R denote the number of constraints in H violated by $x^*(R)$.



We know that $x^*(R) = x^*(T)$, for some $T \subset R$ with $|T| \leq m$.

We want to find $k < n$ such that the probability that $v_R > k$ is less then $1/2$. This probability is bounded above by:

$$\sum_{0 \leq i \leq m} \sum_{T \subset H, |T| = i, v_T > k} \Pr(x^*(T) = x^*(R)),$$

which is no more than:

$$\sum_{0 \leq i \leq m} \binom{n}{i} \frac{\binom{n-i-k}{r-i}}{\binom{n}{r}},$$

ILP Algorithm: Proof of Lemma 2 (cont'd)

which is again no more than:

$$(m+1)\binom{r}{m}\frac{\binom{n-m-k}{r-m}}{\binom{n-m}{r-m}},$$

and using elementary bounds, this quantity is less than $1/2$ for $k \geq 2^{d+1}n(\ln r)/r$.

ILP Algorithm: Running Time

- We have the following **theorem:**

  The ILP algorithm requires expected

  $$O(2^d + 8^d \sqrt{n \ln n} \ln n)$$

  row operations on $O(d^3\phi)$ *-bit vectors, and*

  $$d^{O(d)} \phi \ln n$$

  expected operations on $O(d^{O(1)}\phi)$ *-bit numbers, as* $n \to \infty$ *where the constant factors do not depend on* d *or* ϕ.

Part II: Incremental Linear Programming

Incremental LP

- Randomized incremental algorithms for LP
- Concept:
  - add n constraints in random order,
  - after adding each constraint, determine the optimum of the constraints added so far.
- Two algorithms will be discussed:
  - SeideLP
  - BasisLP

Algorithm SeideLP

Input: A set of constraints H.

Output: The optimum of the LP defined by H.

0. if $|H| = d$, output $\mathcal{B}(H) = H$.

1. Pick a random constraint $h \in H$;

Recursively find $\mathcal{B}(H \setminus \{h\})$;

2.1. if $\mathcal{B}(H \setminus \{h\})$ does not violate h, output $\mathcal{B}(H \setminus \{h\})$ to be the optimum $\mathcal{B}(H)$;

2.2. else project all the constraints of $H \setminus \{h\}$ onto h and recursively solve this new linear programming problem;

SeideLP: Running Time

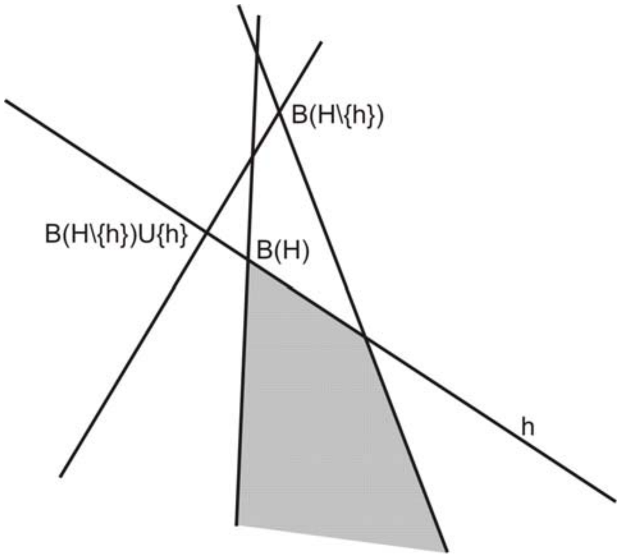
- Let $T(n, d)$ denote an upper bound on the expected running time for a problem with n constraints in d dimensions.
- Then:

$$T(n,d) \leq T(n-1,d) + O(d) + \frac{d}{n}[O(dn) + T(n-1,d-1)].$$

  - *First term:* cost of recursively solving the LP defined by the constraints $H \setminus \{h\}$
  - *Second term:* checking whether h violates $\mathcal{B}(H \setminus \{h\})$
  - *Third term* *(with probability d/n):* cost of projecting + recursively solving smaller LP.
- **Theorem:** *There is a constant b such that the recurrence satisfies the solution* $T(n,d) \leq bnd!$.

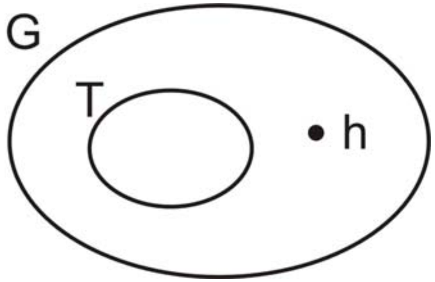
SeideLP: Further Discussion

- In Step 2.2. we completely discard any information obtained from the solution of the LP $H \setminus \{h\}$.



- From the above figure, it follows we must consider all constraints in H.
- But: Can we use $\mathcal{B}(H \setminus \{h\})$ to “jump-start” the recursive call in step 2.2.?
- RESULT: Algorithm BasisLP

Algorithm BasisLP



Input: G, T.

Output: A basis B for G.

0. If $G = T$, output T;

1. Pick a random constraint $h \in G \setminus T$;

$T' =$ **BasisLP**($G \setminus \{h\}, T$);

2.1. if h does not violate T', output T';

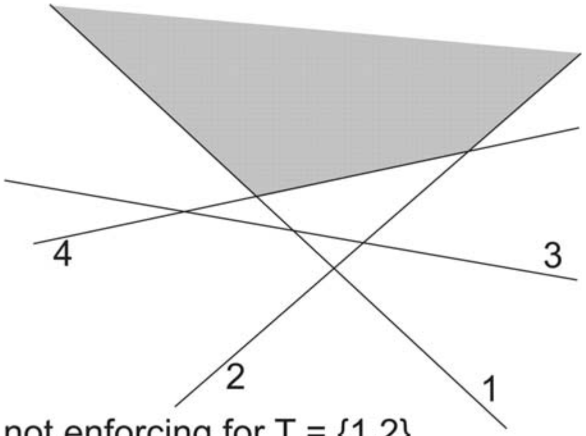
2.2. else output **BasisLP**(G, **Basis**($T' \cup \{h\}$));

Basis returns a basis for a set of $d+1$ or fewer constraints.

BasisLP: Why does it work?

- Each invocation of Basis occurs when the violation test in 2.1. fails (i.e. h does violate T').
- What is the probability that we fail a violation test?
  - Let $|G| = i$,
  - Remember: $h \in G \setminus T$
  - **Pr(** h violates the optimum of $G \setminus \{h\}$ **)** $\leq d/(i - |T|)$
  - This probability decreases further if T contains some of the constraints of $\mathcal{B}(G)$
  - This was indeed the motivation for modifying SeideLP to BasisLP.

BasisLP: Running Time



1 not enforcing for T = {1,2}

- Notation:
  - Given $T \subseteq G \subseteq H$, we call h *enforcing* in (G, T) if $\mathcal{O}(G \setminus \{h\}) < \mathcal{O}(T)$.
  - Let $\Delta_{G,T}$ denote d minus the number of constraints that are enforcing in (G, T). $\Delta_{G,T}$ is called the *hidden dimension* of (G, T).
- **Lemma 1:** *If h is enforcing in (H, T) then (i) $h \in T$, and (ii) h is extreme in all G such that $T \subseteq G \subseteq H$.*
- So, the probability that a violation occurs can be bounded by $\Delta_{G,T}/(i - |T|)$.
- We establish that the $\Delta_{G,T}$ decreases by at least 1 at each recursive call in step 2.2. It turns out $\Delta_{G,T}$ is likely to decrease much faster.
- **Theorem:** *The expected running time of BasisLP is* $O(d^4 2^d n)$.

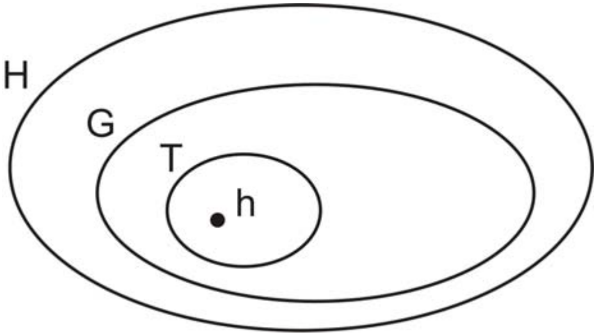
BasisLP: Analysis Details

- *Proof* of Lemma 1. *If h is enforcing in (H,T) then*
  - (i) $h \in T$.

    We have $\mathcal{O}(H \setminus \{h\}) < \mathcal{O}(T)$, which can not be true if T were a subset of $H \setminus \{h\}$.
  - (ii) h *is extreme in all G such that* $T \subseteq G \subseteq H$.

    Assume the contrary: $\mathcal{O}(G \setminus \{h\}) = \mathcal{O}(G)$.

    $\mathcal{O}(T) \leq \mathcal{O}(G) = \mathcal{O}(G \setminus \{h\}) \leq \mathcal{O}(H \setminus \{h\}) < \mathcal{O}(T)$, a contradiction.



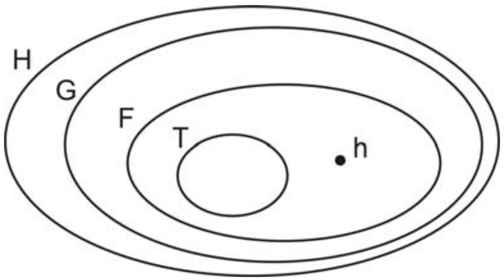
BasisLP: Analysis Details (Cont'd)

- **Lemma 2:** *Let* $T \subseteq F \subseteq G \subseteq H$, *and let* $h \in F \setminus T$ *be an extreme constraint in* F. *Let* S *be a basis of* $\mathcal{B}(F \setminus \{h\}) \cup \{h\}$. *Then:*

  (i) Any constraint g *that is enforcing in* (G, T) *is also enforcing in* (F, S);

  (ii) h *is enforcing in* (F, S);

  (iii) $\Delta_{F,S} \leq \Delta_{G,T} - 1$.



  Proof:
  - (i) $\mathcal{O}(T) \leq \mathcal{O}(F \setminus \{h\}) \leq \mathcal{O}(S)$, $\mathcal{O}(G \setminus \{g\}) < \mathcal{O}(T)$,

    then: $\mathcal{O}(F \setminus \{g\}) \leq \mathcal{O}(G \setminus \{g\}) < \mathcal{O}(T) \leq \mathcal{O}(F \setminus \{h\}) \leq \mathcal{O}(S)$.
  - (ii) Since h is extreme in F, $\mathcal{O}(F \setminus \{h\}) < \mathcal{O}(S)$.
  - (iii) Follows readily.

- So, the numerator of $\Delta_{G,T}/(i - |T|)$, decreases by at least 1 at each execution.

BasisLP: Analysis Details (Cont'd)

- Show that this decrease is likely to be faster.
- Given $T \subseteq F \subseteq G$, and a random $h \in F \setminus T$ we bound the probability that h violates $\mathcal{B}(F \setminus \{h\})$. If it does, check the probability distribution of the resulting hidden dimension.
- **Lemma 3:** *Let $g_1, g_2, \ldots, g_s$ be the extreme constraints of F that are not in T, numbered so that*

  $$\mathcal{O}(F \setminus \{g_1\}) \leq \mathcal{O}(F \setminus \{g_2\}) \leq \ldots$$

  Then, for all l and for $1 \leq j \leq l$, g_j is enforcing in $(F,$ **Basis**$(\mathcal{B}(F \setminus \{g_l\}) \cup \{g_l\}))$. (*proof:* immediate from lemma 2.)
- In other words: when $h = g_l$, then all of $\{g_1, g_2, \ldots, g_l\}$ will be enforcing and the arguments of the recursive call will have hidden dimension $\Delta_{G,T} - l$.
- Observation: since any g_i is equally likely to be h, l is uniformly distributed on the integers in $[1, s]$, and the resulting hidden dimension is uniformly distributed on the integers in $[0, s-1]$.

BasisLP: Analysis Details (Cont'd)

- Let $T(n,k)$ denote the maximum expected number of violation tests for a call to **BasisLP** with arguments (G,T), where $|G|=n$ and $\Delta_{G,T}=k$.
- We get:

$$T(n,k) \le T(n-1,k) + 1 + \frac{T(n,0)+\ldots+T(n,k-1)}{n-d}.$$

- This yields: $T(n,k) \le 2^k(n-d)$,

  and consequently the expected running time of **BasisLP** is $O(d^4 2^d n)$.

Augmenting the analysis with Clarkson's sampling technique improves the running time of the mixed algorithm to $O(d^2 n + b^{\sqrt{d \log d}} \log n)$.