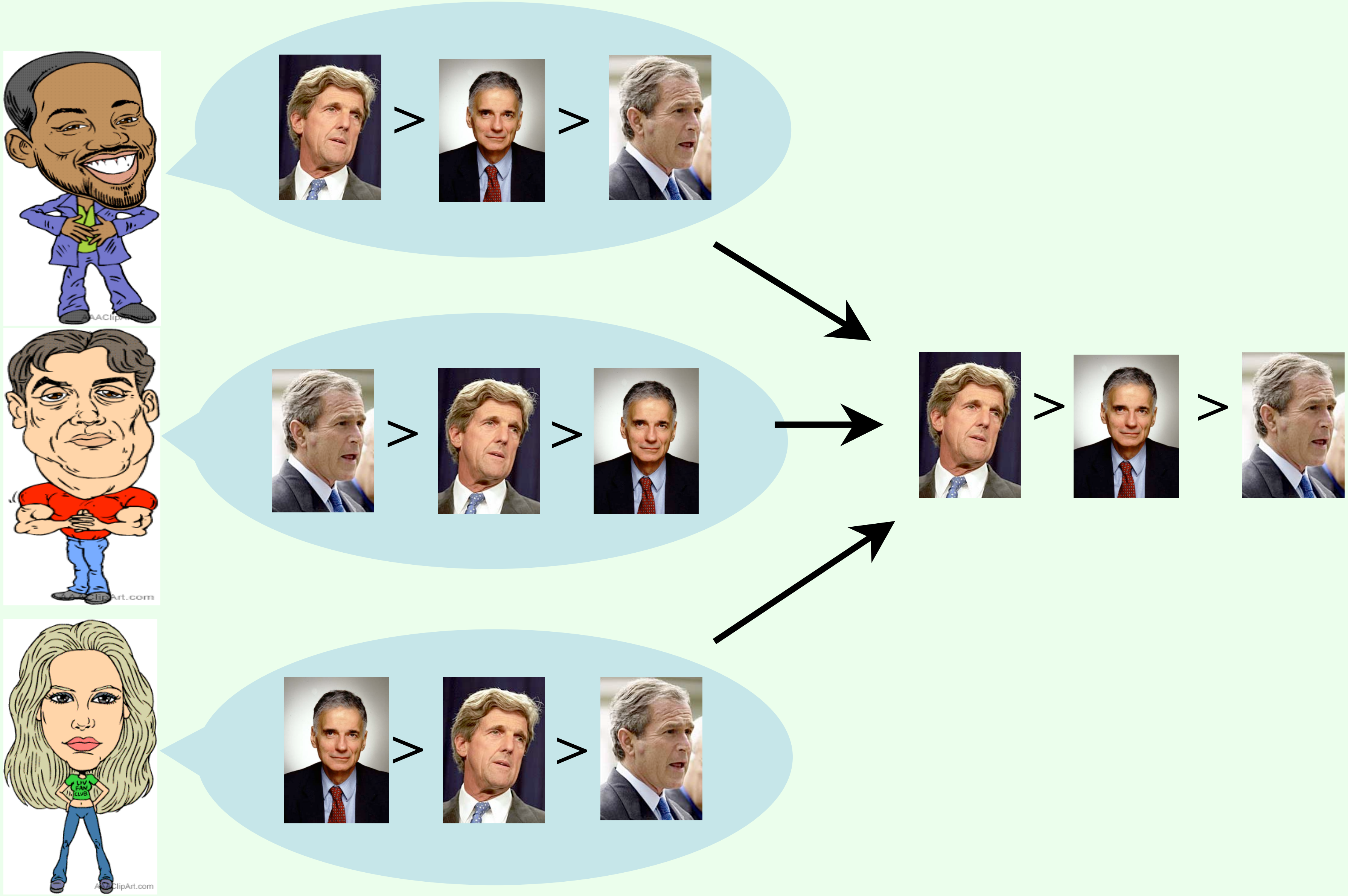
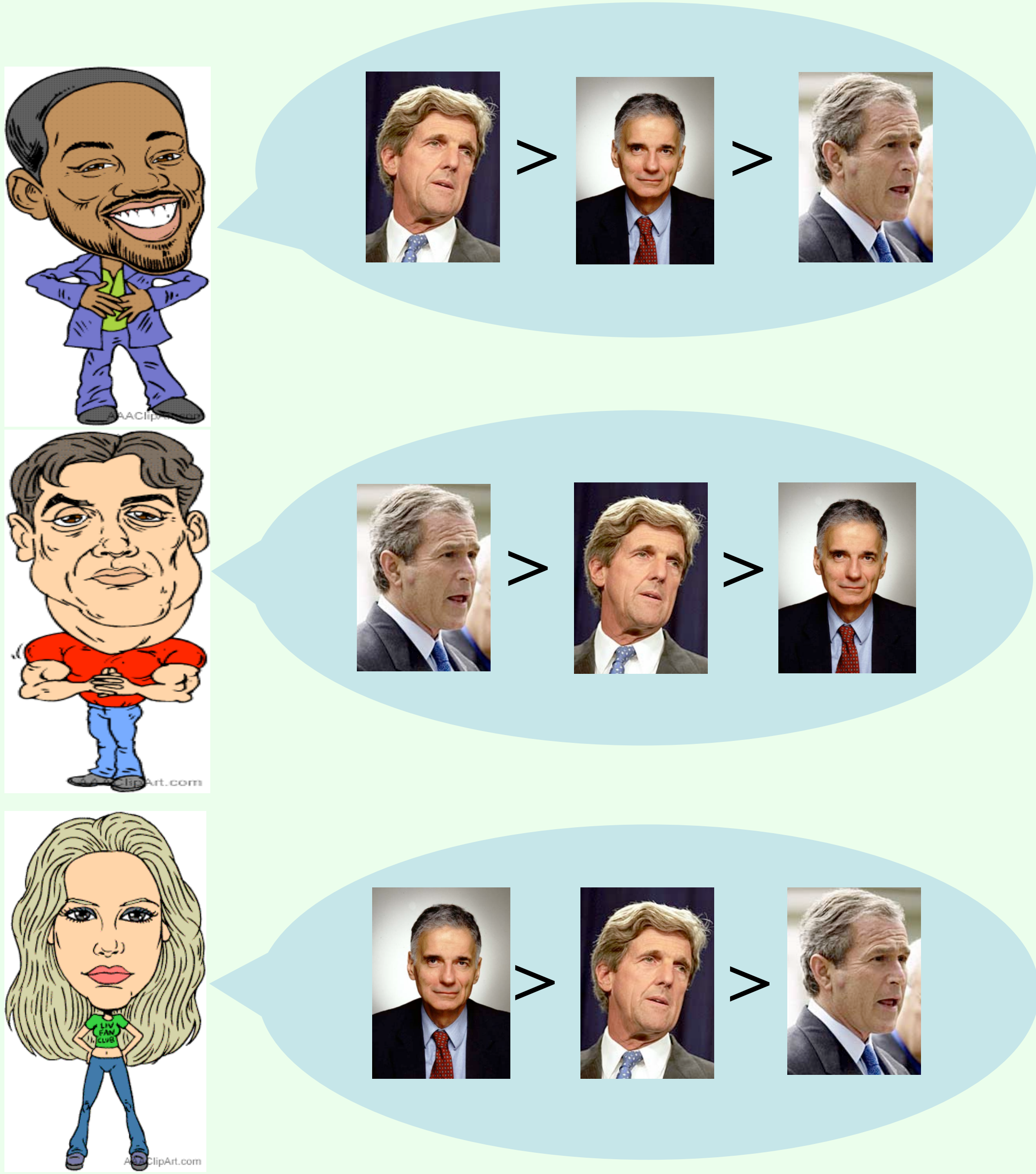
Eliciting Single-Peaked Preferences Using Comparison Queries

Vincent Conitzer
conitzer@cs.duke.edu

Voting



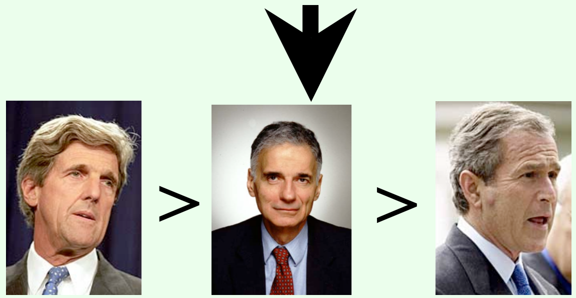
Pairwise elections



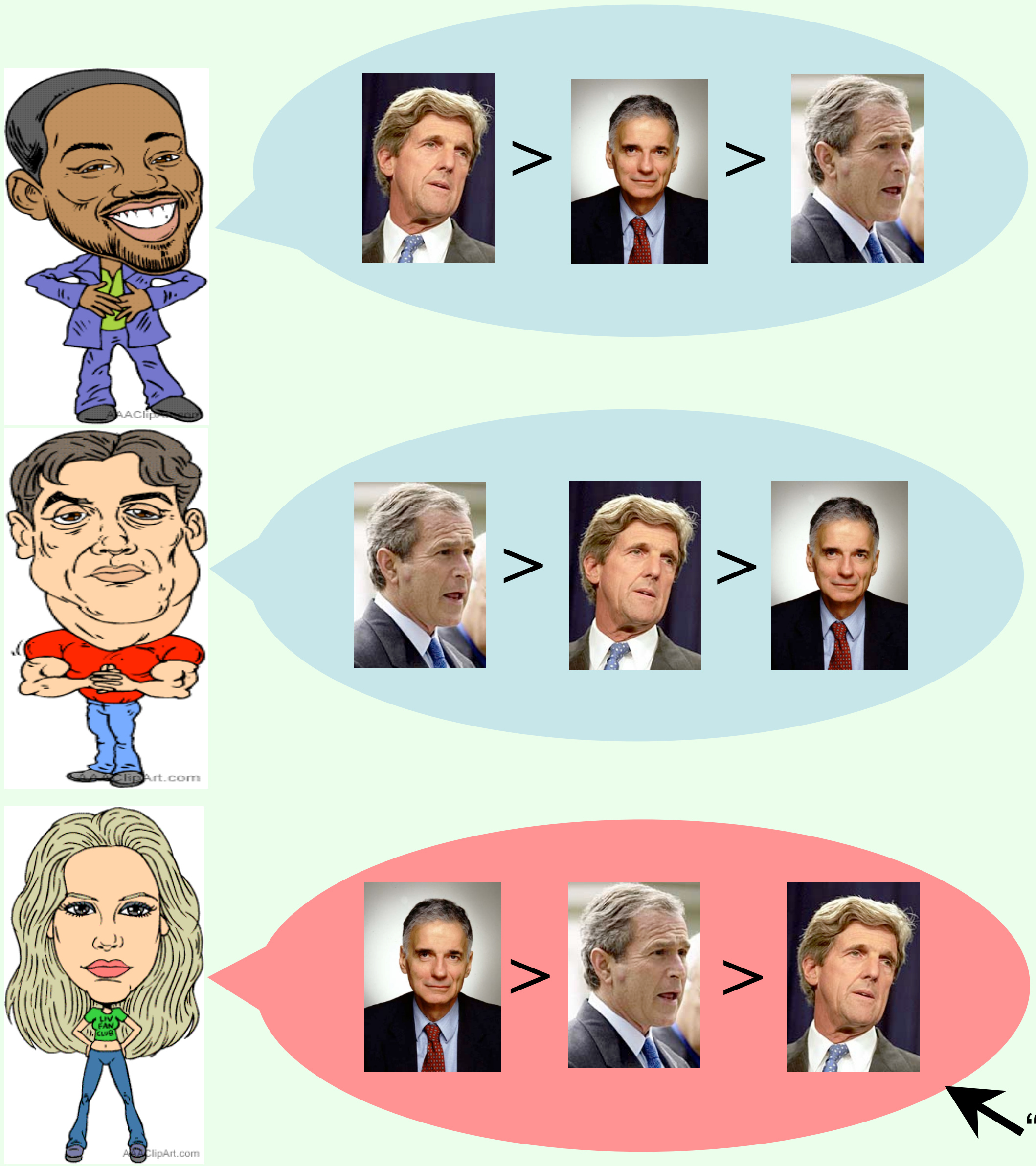
two votes prefer Kerry to Bush

two votes prefer Kerry to Nader

two votes prefer Nader to Bush



Condorcet cycles



two votes prefer Bush to Kerry

two votes prefer Kerry to Nader

two votes prefer Nader to Bush

?

“weird” preferences

Single-peaked preferences [Black 48]

- Suppose alternatives are ordered on a line from left to right (the alternatives' positions)
- E.g. d - b - e - f - a - c
  - Left-wing vs. right-wing political candidates
  - Perhaps the alternatives are numbers, e.g. voting over the size of the budget
  - Voting over locations along a road
  - …
- An agent's preferences are single-peaked with respect to these positions if the agent prefers alternatives that lie closer to her most preferred alternative (on each side)
- f > e > a > d > c > b is not single peaked with respect to above positions: d is ranked above b, but b is closer to f and on the same side as d
- f > e > b > a > c > d is single-peaked

Nice properties of single-peaked preferences

- Suppose every voter's preferences are single-peaked (with respect to the same positions for the alternatives)
- If a wins the pairwise election between a and b,
- and b wins the pairwise election between b and c,
- then a must win the pairwise election between a and c
  - I.e. no Condorcet cycles
- So we can use pairwise elections to determine the ranking
- This is also strategy-proof
- (Gibbard-Satterthwaite theorem: for general preferences, no reasonable deterministic voting rule is strategy-proof)

Preference elicitation

- Direct mechanisms ask each agent to reveal complete preferences
  - In voting, each agent gives an entire ranking
- Can be cumbersome to agents
  - Have to decide and communicate entire preferences without any help
  - Especially hard if there are many alternatives
- In preference elicitation, the center (elicitor) repeatedly asks agents "natural" queries about their preferences
  - E.g. comparison queries: do you prefer a to b?
- In this paper, the elicitor wants to learn each agent's complete preferences, using comparison queries
- How many queries are needed?

Eliciting general preferences
(not single-peaked)

- Discover the full ranking > of the m alternatives based on comparison queries
- Equivalent to sorting a list of m elements using only binary comparisons
- E.g. MergeSort algorithm solves this with $O(m \log m)$ queries
- Any algorithm is $\Omega(m \log m)$

- With n voters, many voting rules require $\Omega(nm \log m)$ communication even just to determine the winner [Conitzer & Sandholm EC05]

Eliciting preferences given positions

- Voter’s preferences: b > c > e > f > a > d (unknown)
- Positions: e - c - b - f - a - d (known)
- Let us find the most preferred alternative first
- “b > f?” “Yes”
  - Tells us that most preferred alternative must be e, c, b
  - ~ binary search
- “c > b?” “No”
  - So b must be most preferred
  - Next-ranked alternative must be c or f
- “c > f?” “Yes”
  - Next-ranked alternative must be e or f
- “e > f?” “Yes”
  - Now we know the ranking must be b > c > e > f > a > d

How many queries does this take?

- Finding the most preferred alternative takes at most 1+ log m queries
  - Binary search
- The remainder will require at most m - 2 queries
  - Each query allows us to add the next alternative to the ranking

What if we do not know the positions?

- Any preferences are single-peaked with respect to some positions
- E.g. f > e > a > d > c > b is consistent with respect to
  - f - e - a - d - c - b
  - d - e - f - a - c - b
  - many other positionings
- So eliciting the first voter's preferences will require $\Omega(m \log m)$ queries
- Once we know one voter's preferences, we know something about the positions
- Will show that this is enough information to need only $O(m)$ queries for next voter

Eliciting preferences using another voter's preferences (stage 1)

- Positions: e - c - b - f - a - d (unknown)
- Current voter's preferences: c > e > b > f > a > d (unknown)
- Previous voter's preferences: a > d > f > b > c > e (known)
- Let us find the most preferred alternative first
- Cannot use binary search this time, just do one at a time
- "a > d?" "Yes"
- "a > f?" "No"
- "f > b?" "No"
- "b > c?" "No"
- "c > e?" "Yes"
- So most preferred alternative is c

Eliciting preferences using another voter's preferences (stage 2)

- Positions: e - c - b - f - a - d (unknown)
- Current voter's preferences: c > e > b > f > a > d (unknown)
- Previous voter's preferences: a > d > f > b > c > e (known)
- Let us find out which alternatives lie between a (previous voter's most preferred) and c (current voter's most preferred) in the positions
- Previous voter must prefer such alternatives to c
  - Could be d, f, b
- Current voter must prefer such alternatives to a
  - "d > a?" "No"
  - "f > a?" "Yes"
  - "b > a?" "Yes"
- So b and f lie between a and c
- Current voter's preferences over a, c, b, f must be opposite of previous voter's, i.e. c > b > f > a

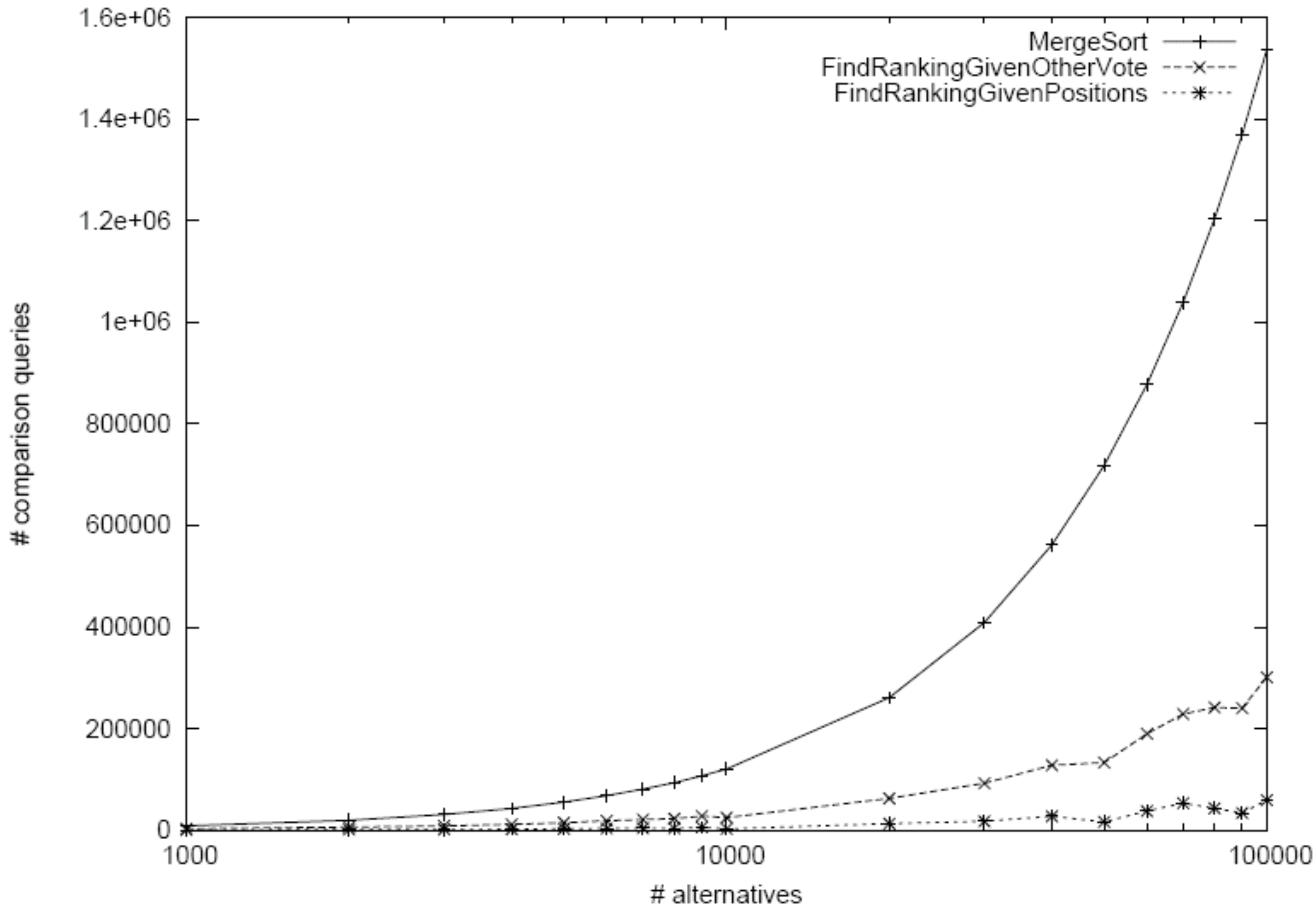
Eliciting preferences using another voter's preferences (stage 3)

- Positions: e - c - b - f - a - d (unknown)
- Current voter's preferences: c > e > b > f > a > d (unknown)
- Previous voter's preferences: a > d > f > b > c > e (known)
- We know c > b > f > a; must integrate d and e
  - In order of previous voter's preferences, i.e. d before e
- Start by comparing to currently last-ranked alternative
- "d > a?" "No"
  - Now we know c > b > f > a > d
- "e > d?" "Yes"
  - e must lie on opposite side from d in positions, since known and current voters disagree on ranking of e and d
  - Start from the top…
- "e > b?" "Yes"
  - Now we know c > e > b > f > a > d

How many queries does this take?

- Finding the most preferred alternative (stage 1) takes at most m -1 queries
- Finding the alternatives between the previous and current voter's most preferred alternatives (stage 2) takes at most m - 2 queries
- Integrating the remaining alternatives (stage 3) requires at most 2m - 3 queries
  - More complex argument
  - Requires keeping track of the worst-ranked alternative above which we will never insert another alternative
- Total upper bound is 4m - 6 queries

Experimental results



Conclusions

- Determining general preferences requires $\Omega(m \log m)$ comparison queries
- If preferences are single-peaked and
  - the positions of the alternatives are known, or
  - at least one other voter's preferences are known,
- then preferences can be elicited using $O(m)$ queries
  - There is also an $\Omega(m)$ lower bound
- What about more general families of preferences?
  - E.g. alternatives take positions on the plane rather than the line
  - Many of the nice properties go away…

Thank you for your attention!