Filtering with the Crowd

LRI

Benoît Groz, Ezra Levin, Isaco Meiljson, Tova Milo

Tel-Aviv University
Univ. Paris Saclay

15 Mars 2016

Outline

Broader perspective: CrowdSourcing project at TAU

CrowdSourcing: engaging web users to contribute and process information.

Use cases: Wikipedia, annotations for ML data (images,video, text processing), reviews, data cleaning...

FP7 MODAS project (T. Milo et al.):
develop foundations for the management of large-scale crowd-sourced data.

Interface	Mining with the crowd	Query optimization
(NLP, query refining)	*(ontologies, association rules, data cleaning)*	*(Planning queries, Filtering, Skyline)*

...

Filtering with the Crowd

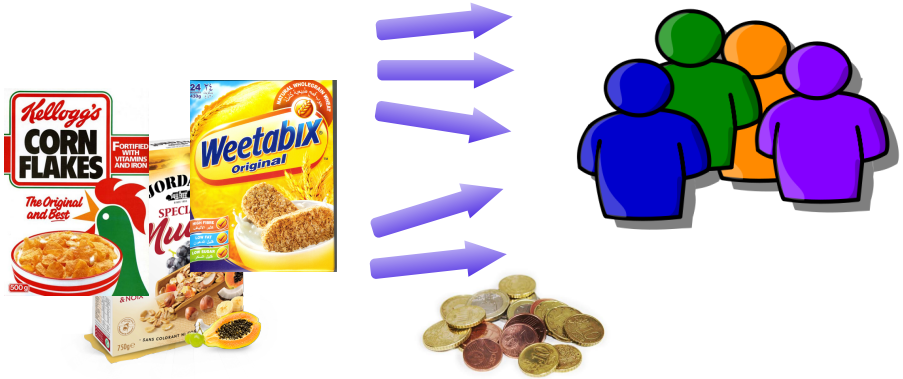
$s = 50\%$ gluten-free cereals
$e_0 = e_1 = 40\%$ errors

Filtering in CrowdScreen's model
Select with minimum number of tasks

- all cereals with gluten
- with $\leq \tau = 10\%$ of misclassification

Compute sequential test in terms of e_0, e_1, s, τ and budget m.

redundant tasks



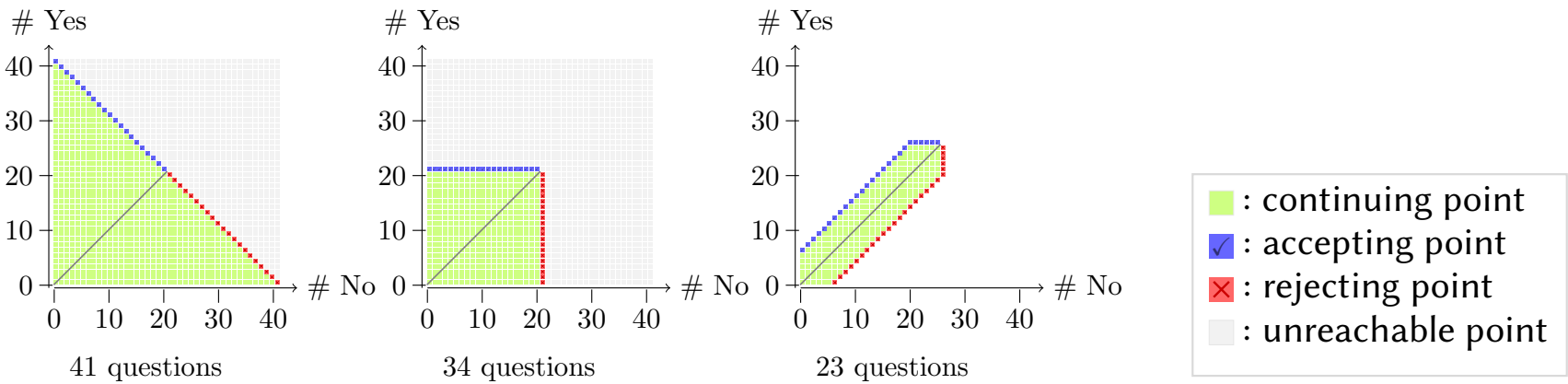
Aim: minimize cost=#tasks

Strong assumption: s, e_0, e_1 known in advance.
($\hookrightarrow$ may use sampling)

Strategies

Compute strategy when

- 50% of cereals contain gluten
- error probability 0.4 per answer
- we wish at most 10% error.
- we can afford at most m (say, 51) questions per cereal.



Minimize expected cost for given error threshold and budget

Here:

error rates $e_0, e_1 = 0.4$, selectivity $s = 0.5$, error threshold $\tau = 0.1$, budget $m = 51$

In general $s \neq .5$ and $e_0 \neq e_1$ but similar shape…

Seems a hard problem...

Problem: computing optimal strategy.
(i.e., optimal stopping time in a sequential test)

Complexity bounds

- Check all possible strategies: $O(2^{m^2})$
- Check all ladder strategies: $O(2^{2m})$

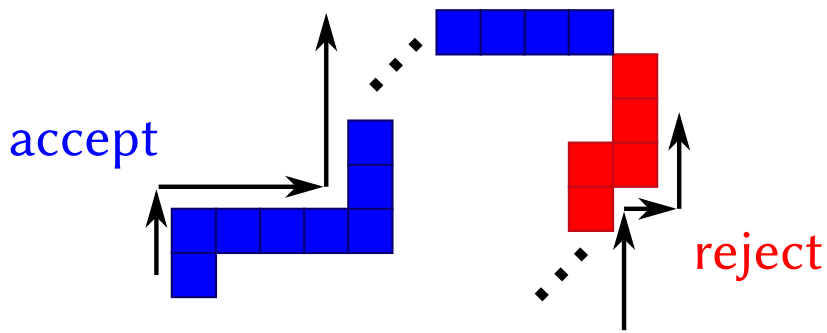


Figure: ladder strategy

⇒ Heuristics
⇒ Probabilistic relaxation.

Contributions: outline

Crowdscreen [Parameswaran et al, SIGMOD'12]

- Defined the framework
- A linear program to compute the optimal probabilistic strategy
- Two gradient-based heuristics in $O(m^5)$; `shrink` and `growth`

Our contributions

- Complexity Analysis and show that algorithms scale poorly
- Improve the complexity of both `growth` and `shrink` to $O(m^4)$, and remedy a "flaw" in `growth`
- Propose a scalable heuristic based on the well-known SPRT
- Establish connections between probabilistic and deterministic strategies.

Outline

Panel of algorithms

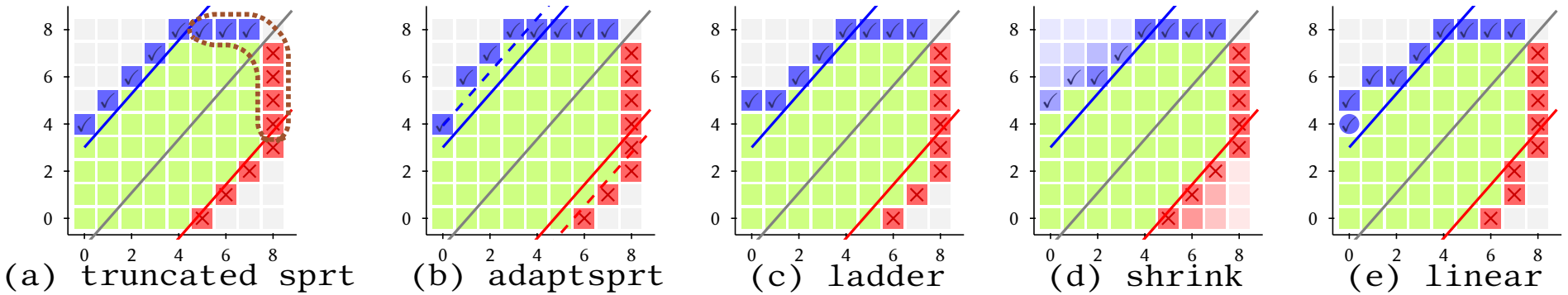


Figure: Strategies returned for $e_0 = .25$, $e_1 = .2$, $s = .8$, $\tau = .0075$, and $m = 15$.

: continuing point
: accepting point
: rejecting point
: $P_{stop} = .62$
: unreachable point
: decision line
: SPRT lines

SPRT[Wald'44]: stops when error $< \tau$
$\Leftrightarrow$ likelihood ratio $LR(x, y) \notin [\frac{\tau}{1-\tau}, \frac{1-\tau}{\tau}]$

$$LR(x, y) = \frac{\Pr(\text{reach}(x, y) \mid \text{gluten})}{\Pr(\text{reach}(x, y) \mid \text{gluten-free})} \times \frac{s}{1 - s}$$

$$\log(LR(x, y)) = \log \frac{s}{1-s} + x \log\left(\frac{e_1}{1-e_0}\right) + y \log\left(\frac{1-e_1}{e_0}\right)$$

Panel of algorithms

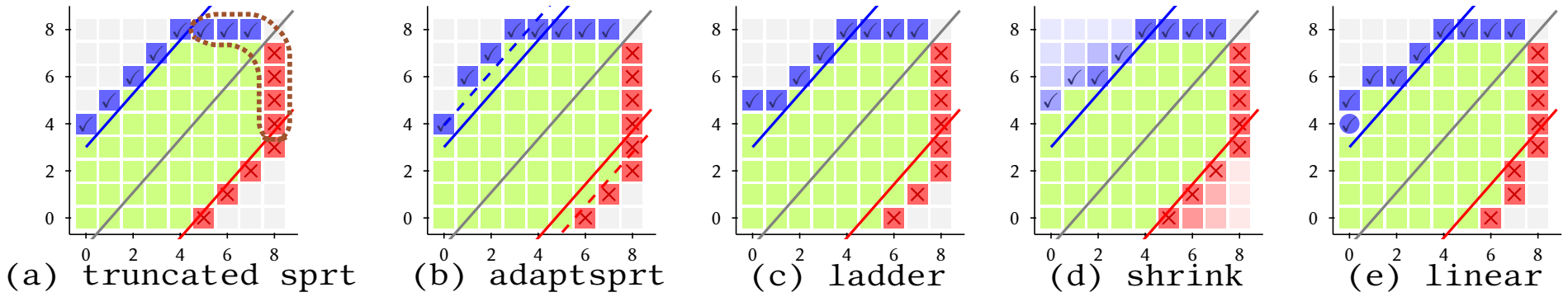


Figure: Strategies returned for $e_0 = .25$, $e_1 = .2$, $s = .8$, $\tau = .0075$, and $m = 15$.

: continuing point
: accepting point
: rejecting point
: $P_{stop} = .62$
: unreachable point
: decision line
: SPRT lines

Truncated SPRT: stops when error$< \tau$ or exceed m
Truncation may raise error above τ.
$\hookrightarrow$ using binary search we can compute the optimal LR threshold to obtain expected error $< \tau$.

Panel of algorithms

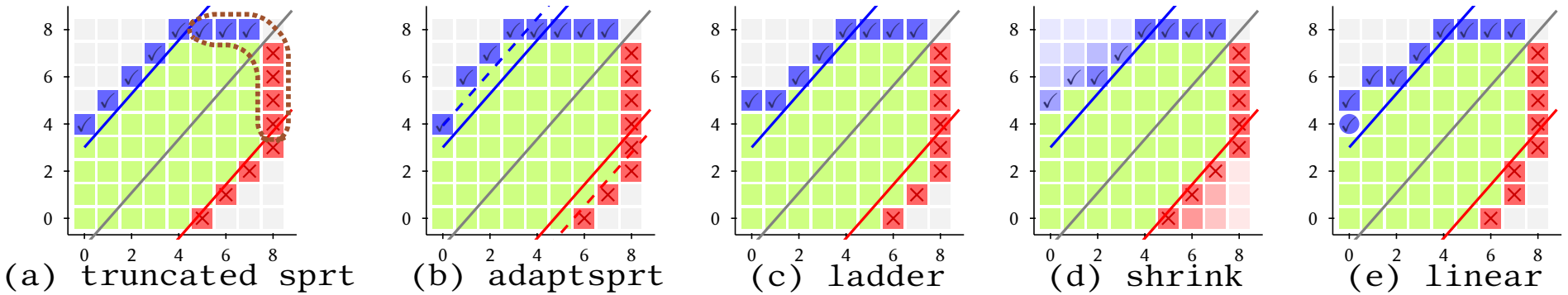
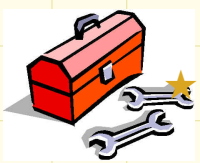


Figure: Strategies returned for $e_0 = .25$, $e_1 = .2$, $s = .8$, $\tau = .0075$, and $m = 15$.

: continuing point
: accepting point
: rejecting point
: $P_{stop} = .62$
: unreachable point
: decision line
: SPRT lines

Number of `ladder` strategies: $O(2^{2m}/m)$
(a little fewer, but $\Omega(2^{2m}/m^3)$)
Error and Cost computed in amortized $O(m)$.



Incremental evaluation/enumeration[Knuth]

Panel of algorithms

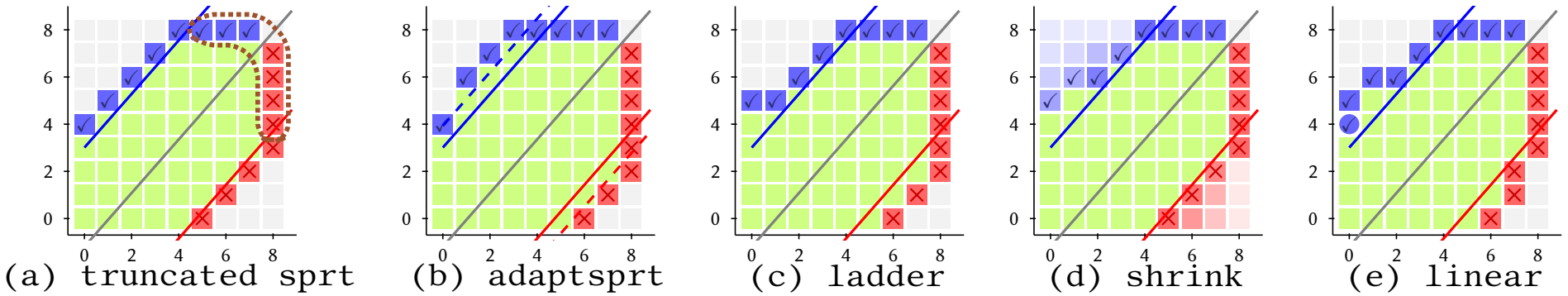


Figure: Strategies returned for $e_0 = .25$, $e_1 = .2$, $s = .8$, $\tau = .0075$, and $m = 15$.

: continuing point
: accepting point
: rejecting point
: $P_{stop} = .62$
: unreachable point
: decision line
: SPRT lines

shrink: gradient-like heuristic

For each (x, y) compute $-\frac{\Delta C}{\Delta E} = -\frac{C'-C}{E'-E} \geq 0$.
Add terminating point at maximum with $E' \leq \tau$.

Proposition

We can compute all ratios $-\frac{\Delta C}{\Delta E}(x, y)$ in $O(m^2)$.

Panel of algorithms

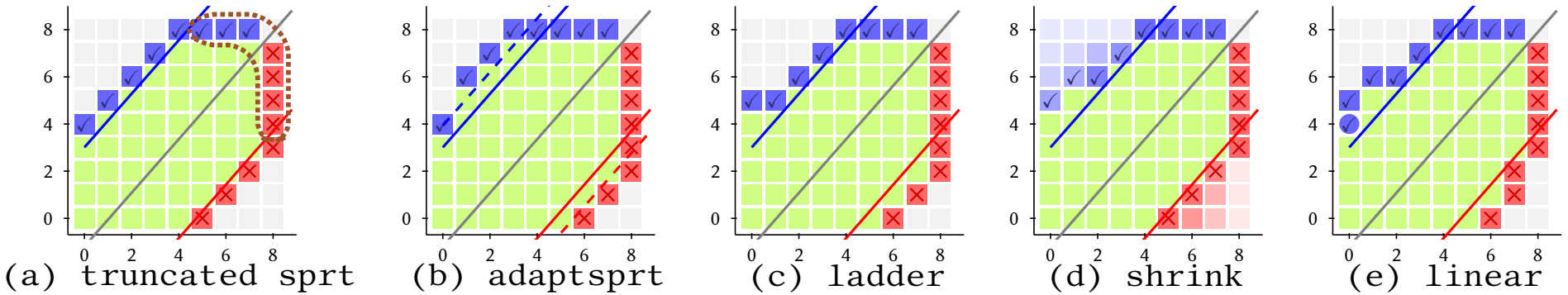


Figure: Strategies returned for $e_0 = .25$, $e_1 = .2$, $s = .8$, $\tau = .0075$, and $m = 15$.

: continuing point
: accepting point
: rejecting point
: $P_{stop} = .62$
: unreachable point
: decision line
: SPRT lines

Probabilistic strategy:

$\Rightarrow$ lower cost& linear program for optimal strategy

Theorem

- There is an optimal strategy with a single probabilistic point (unique in general).
- Instead of linear program, we can use `shrink` with probabilistic point.

Outline

Panel of algorithms

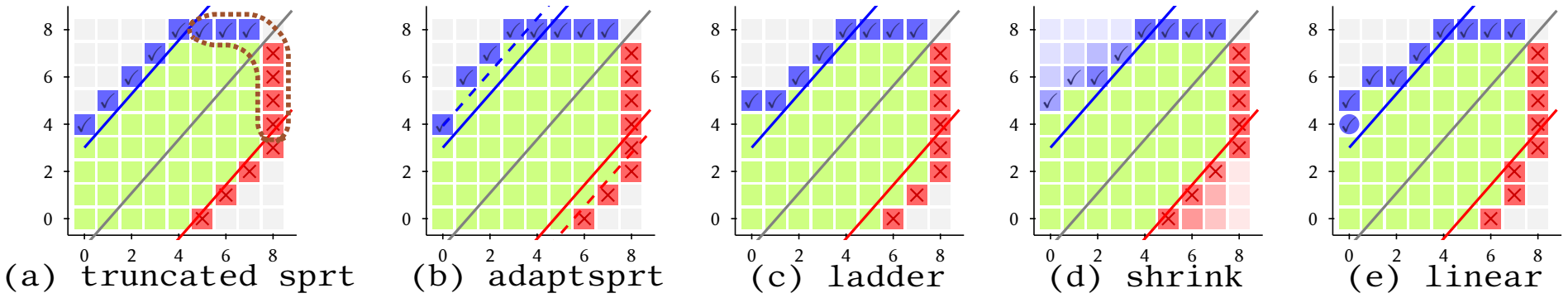


Figure: Strategies returned for $e_0 = .25$, $e_1 = .2$, $s = .8$, $\tau = .0075$, and $m = 15$.

	SPRT	AdaptSprt	ladder	shrink	linear
cost	6.94	7.748	7.59	7.73	7.56
error	0.008	0.00741	0.00749	0.00748	.0075

: continuing point
: accepting point
: rejecting point
: $P_{stop} = .62$
: unreachable point
: decision line
: SPRT lines

Experiments on real Crowd

	question	s	e_0	e_1
Q1	photos from Australia	.18	.25	.36
Q2	photos from Greece or Cyprus	.26	.27	.32
Q3	dishes containing dairy	.17	.11	.27
Q4	dishes containing onions	.54	.38	.27
~~Q5~~	~~dishes containing garlic~~	~~.62~~	~~.44~~	~~.48~~
~~Q6~~	~~dishes containing eggs~~	~~.19~~	~~.22~~	~~.57~~

Figure: Question parameters

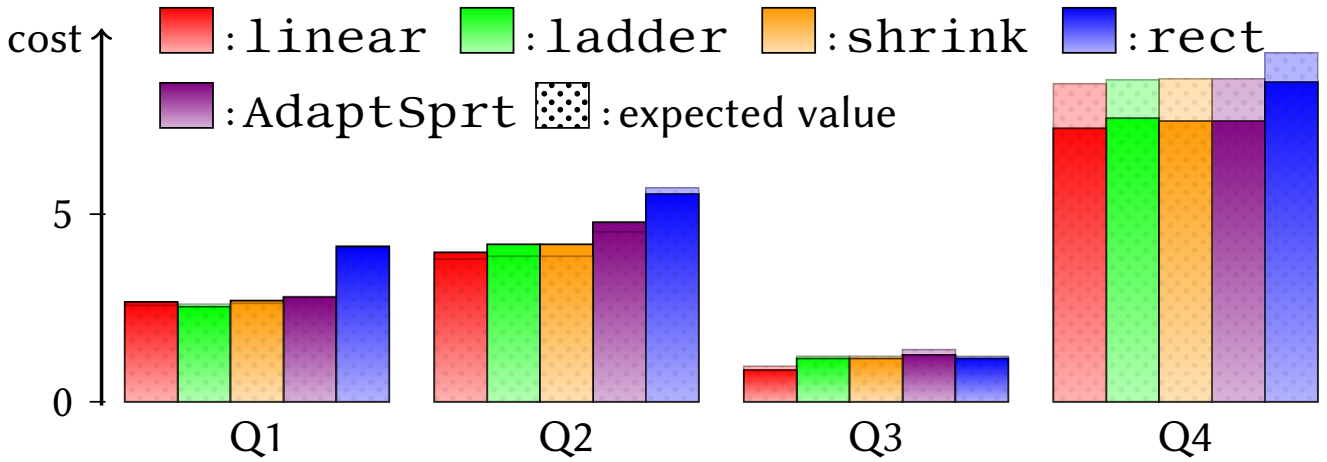


Figure: Average cost per item (with $m = 12$, $\tau = .1$)

Running time

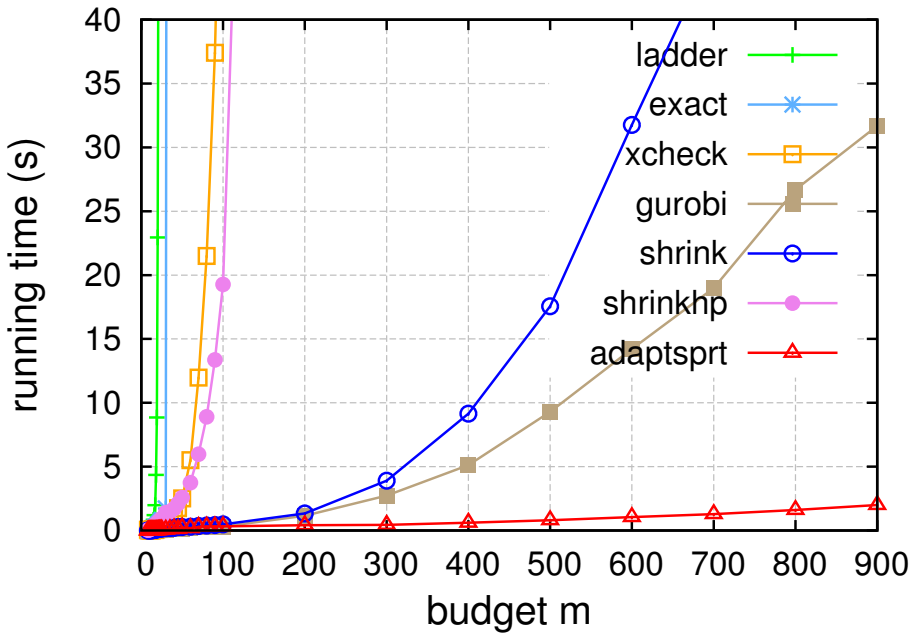


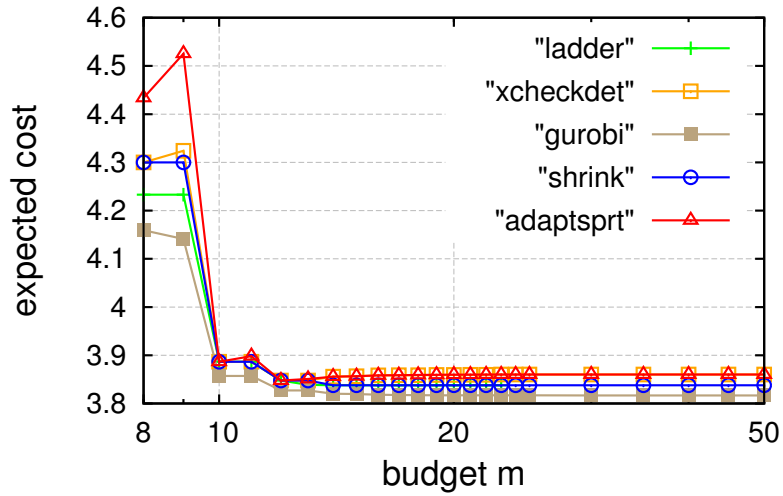
Figure: Average running time of algorithms on random instances

(e_0, e_1, τ, m, s)	`ladder` (PyPy)	naive (PyPy)	naive (cPython)
(.25, .2, .0075, 15, .8)	0.8s	6.4s	3min
(.2, .25, .05, 18, .6)	5s	4.5min	2.7h

(e_0, e_1, τ, m, s)	`shrink` (PyPy)	naive (PyPy)	naive (cPython)
(.25, .2, .0075, 40, .8)	.24s	.6s	23s
(.25, .2, .0075, 200, .8)	1.5s	12 min	> 5h

Figure: Running time for `ladder` and `shrink` variants

Sensitivity of the strategy ($\tau = .1, m = 12$)



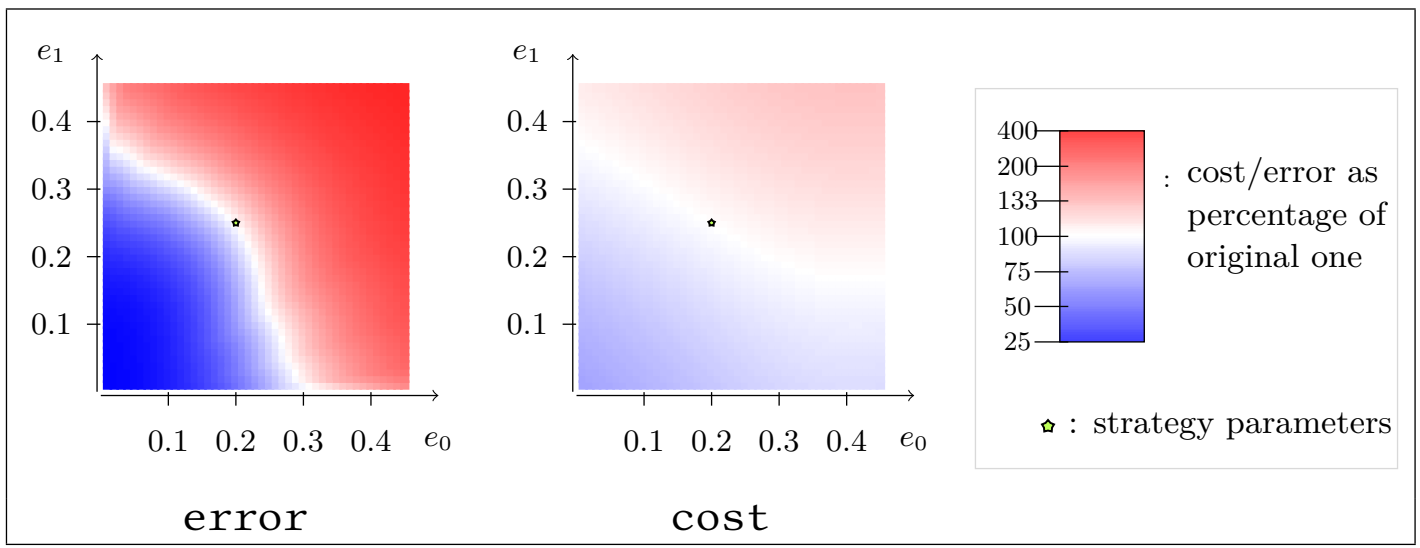


Figure: For $e_0 = .2$, $e_1 = .25$, $\tau = .05$, $s = .6$: cost, and sensitivity (only `shrink`, $m = 15$).

Conclusion

CrowdScreeen's purpose: classify multiple items according to predefined strategy.
Problem: compute "optimal" stopping rules given parameters.

Our contributions:

- ✓ optimize previous algorithms
- ✓ explain or fix properties observed in original framework
- ✓ establish connection between `shrink` and probabilistic strategy
- ✓ experimental evaluation