Ranking Distributed Probabilistic Data

Feifei Li

Ke Yi

Jeffrey Jestes

Introduction

- Ranking queries are important tools used to return only the most significant results

Introduction

- Ranking queries are important tools used to return only the most significant results
- Ranking queries are arguably one of the most important tools for distributed applications

Introduction

- Ranking queries are important tools used to return only the most significant results
- Ranking queries are arguably one of the most important tools for distributed applications
- Not surprisingly, many distributed applications such as sensor networks with fuzzy measurements are also inherently uncertain in nature

Introduction

- Ranking queries are important tools used to return only the most significant results
- Ranking queries are arguably one of the most important tools for distributed applications
- Not surprisingly, many distributed applications such as sensor networks with fuzzy measurements are also inherently uncertain in nature
- Such applications may be best represented with probabilistic data

Introduction

- Ranking queries are important tools used to return only the most significant results
- Ranking queries are arguably one of the most important tools for distributed applications
- Not surprisingly, many distributed applications such as sensor networks with fuzzy measurements are also inherently uncertain in nature
- Such applications may be best represented with probabilistic data
- Even though distributed probabilistic data is relatively common, there has been no prior research on how to rank distributed probabilistic data

Attribute-Level Model of Uncertainty (with a scoring attribute)

tuples	score
t_1	$X_1 = \{(v_{1,1}, p_{1,1}), (v_{1,2}, p_{1,2}), \ldots, (v_{1,b_1}, p_{1,b_1})\}$
t_2	$X_2 = \{(v_{2,1}, p_{2,1}), \ldots, v_{2,b_2}, p_{2,b_2})\}$
$\vdots$	$\vdots$
t_N	$X_N = \{(v_{N,1}, p_{N,1}), \ldots, (v_{N,b_N}, p_{N,b_N})\}$

Attribute-Level Model of Uncertainty (with a scoring attribute)

tuples	score
t_1	$X_1 = \{(v_{1,1}, p_{1,1}), (v_{1,2}, p_{1,2}), \ldots, (v_{1,b_1}, p_{1,b_1})\}$
t_2	$X_2 = \{(v_{2,1}, p_{2,1}), \ldots, v_{2,b_2}, p_{2,b_2})\}$
$\vdots$	$\vdots$
t_N	$X_N = \{(v_{N,1}, p_{N,1}), \ldots, (v_{N,b_N}, p_{N,b_N})\}$

Attribute-Level Model of Uncertainty (with a scoring attribute)

tuples	score
t_1	$X_1 = \{(v_{1,1}, p_{1,1}), (v_{1,2}, p_{1,2}), \ldots, (v_{1,b_1}, p_{1,b_1})\}$
t_2	$X_2 = \{(v_{2,1}, p_{2,1}), \ldots, v_{2,b_2}, p_{2,b_2})\}$
$\vdots$	$\vdots$
t_N	$X_N = \{(v_{N,1}, p_{N,1}), \ldots, (v_{N,b_N}, p_{N,b_N})\}$

Example Attribute-Level Uncertain Database

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$

world W	$\Pr[W]$
$\{t_1 = 120, t_2 = 103, t_3 = 98\}$	$0.8 \times 0.7 \times 1 = 0.56$
$\{t_1 = 120, t_3 = 98, t_2 = 70\}$	$0.8 \times 0.3 \times 1 = 0.24$
$\{t_2 = 103, t_3 = 98, t_1 = 62\}$	$0.2 \times 0.7 \times 1 = 0.14$
$\{t_3 = 98, t_2 = 70, t_1 = 62\}$	$0.2 \times 0.3 \times 1 = 0.06$

Example Attribute-Level Uncertain Database

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$

world W	$\Pr[W]$
$\{t_1 = 120, t_2 = 103, t_3 = 98\}$	$0.8 \times 0.7 \times 1 = 0.56$
$\{t_1 = 120, t_3 = 98, t_2 = 70\}$	$0.8 \times 0.3 \times 1 = 0.24$
$\{t_2 = 103, t_3 = 98, t_1 = 62\}$	$0.2 \times 0.7 \times 1 = 0.14$
$\{t_3 = 98, t_2 = 70, t_1 = 62\}$	$0.2 \times 0.3 \times 1 = 0.06$

Example Attribute-Level Uncertain Database

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$

world W	$\Pr[W]$
$\{t_1 = 120, t_2 = 103, t_3 = 98\}$	$0.8 \times 0.7 \times 1 = 0.56$
$\{t_1 = 120, t_3 = 98, t_2 = 70\}$	$0.8 \times 0.3 \times 1 = 0.24$
$\{t_2 = 103, t_3 = 98, t_1 = 62\}$	$0.2 \times 0.7 \times 1 = 0.14$
$\{t_3 = 98, t_2 = 70, t_1 = 62\}$	$0.2 \times 0.3 \times 1 = 0.06$

Example Attribute-Level Uncertain Database

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$

world W	$\Pr[W]$
$\{t_1 = 120, t_2 = 103, t_3 = 98\}$	$0.8 \times 0.7 \times 1 = 0.56$
$\{t_1 = 120, t_3 = 98, t_2 = 70\}$	$0.8 \times 0.3 \times 1 = 0.24$
$\{t_2 = 103, t_3 = 98, t_1 = 62\}$	$0.2 \times 0.7 \times 1 = 0.14$
$\{t_3 = 98, t_2 = 70, t_1 = 62\}$	$0.2 \times 0.3 \times 1 = 0.06$

Example Attribute-Level Uncertain Database

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$

world W	$\Pr[W]$
$\{t_1 = 120, t_2 = 103, t_3 = 98\}$	$0.8 \times 0.7 \times 1 = 0.56$
$\{t_1 = 120, t_3 = 98, t_2 = 70\}$	$0.8 \times 0.3 \times 1 = 0.24$
$\{t_2 = 103, t_3 = 98, t_1 = 62\}$	$0.2 \times 0.7 \times 1 = 0.14$
$\{t_3 = 98, t_2 = 70, t_1 = 62\}$	$0.2 \times 0.3 \times 1 = 0.06$

Example Attribute-Level Uncertain Database

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$

world W	$\Pr[W]$
$\{t_1 = 120, t_2 = 103, t_3 = 98\}$	$0.8 \times 0.7 \times 1 = 0.56$
$\{t_1 = 120, t_3 = 98, t_2 = 70\}$	$0.8 \times 0.3 \times 1 = 0.24$
$\{t_2 = 103, t_3 = 98, t_1 = 62\}$	$0.2 \times 0.7 \times 1 = 0.14$
$\{t_3 = 98, t_2 = 70, t_1 = 62\}$	$0.2 \times 0.3 \times 1 = 0.06$

Ranking (top-k) queries (with scores)

- Very useful queries: rank by importance, rank by similarity, rank by relevance, k-nearest neighbors

Ranking (top-k) queries (with scores)

- Very useful queries: rank by importance, rank by similarity, rank by relevance, k-nearest neighbors
  - U-top$k$: [Soliman, Ilyas, Chang, 07], [Yi, Li, Srivastava, Kollios, 08]
  - U-kRanks: [Soliman, Ilyas, Chang, 07], [Lian, Chen, 08], [Yi, Li, Srivastava, Kollios, 08]
  - PT-k: [Hua, Pei, Zhang, Lin, 08]
  - Global-top$k$: [Zhang, Chomicki, 08]

Ranking (top-k) queries (with scores)

- Very useful queries: rank by importance, rank by similarity, rank by relevance, k-nearest neighbors
  - U-top$k$: [Soliman, Ilyas, Chang, 07], [Yi, Li, Srivastava, Kollios, 08]
  - U-kRanks: [Soliman, Ilyas, Chang, 07], [Lian, Chen, 08], [Yi, Li, Srivastava, Kollios, 08]
  - PT-k: [Hua, Pei, Zhang, Lin, 08]
  - Global-top$k$: [Zhang, Chomicki, 08]
  - Expected ranks: [Cormode, Li, Yi, 09]

Ranking Query Properties

Ranking method	*Exact-k*	*Containment*	*Unique-Rank*	*Value-Invariant*	*Stability*
U-top$k$	weak	×	✓	✓	✓
U-kRanks	✓	✓	×	✓	×
PT-k	×	weak	✓	✓	✓
Global-top$k$	✓	×	✓	✓	✓
Expected Ranks	✓	✓	✓	✓	✓

- [Cormode, Li, Yi, 09] has proven that the Expected Ranks definition satisfies all of the above properties while no other definition does

Expected Ranks

- We can see a tuple t's **rank distribution** as a discrete distribution consisting of pairs of $(rank_W(t), Pr[W])$ for all possible worlds W where $rank_W(t)$ is the rank of t in W

Expected Ranks

- We can see a tuple t's **rank distribution** as a discrete distribution consisting of pairs of $(rank_W(t), Pr[W])$ for all possible worlds W where $rank_W(t)$ is the rank of t in W
- The expectance of a distribution is an important statistical property and can provides us important information about a tuple's **rank distribution**

Expected Ranks

- We can see a tuple t's **rank distribution** as a discrete distribution consisting of pairs of $(rank_W(t), Pr[W])$ for all possible worlds W where $rank_W(t)$ is the rank of t in W
- The expectance of a distribution is an important statistical property and can provides us important information about a tuple's **rank distribution**
- Formally, the expected rank of a tuple t_i, $r(t_i)$, may be defined as

$$r(t_i) = \sum_{W \in \mathcal{W}} Pr[W] \times rank_W(t_i) \tag{1}$$

where,

$rank_W(t_i) = |\{t_j \in W | w_{t_j} > w_{t_i}\}|$

w_{t_i} = score attribute value of t_i in W

$\mathcal{W}$ = the set of all possible W

Expected Ranks Example

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$

world W	$\Pr[W]$
$\{t_1 = 120, t_2 = 103, t_3 = 98\}$	$0.8 \times 0.7 \times 1 = 0.56$
$\{t_1 = 120, t_3 = 98, t_2 = 70\}$	$0.8 \times 0.3 \times 1 = 0.24$
$\{t_2 = 103, t_3 = 98, t_1 = 62\}$	$0.2 \times 0.7 \times 1 = 0.14$
$\{t_3 = 98, t_2 = 70, t_1 = 62\}$	$0.2 \times 0.3 \times 1 = 0.06$

tuple	r(tuple)
t_1	$0.56 \times 0 + 0.24 \times 0 + 0.14 \times 2 + 0.06 \times 2 = 0.4$
t_2	$0.56 \times 1 + 0.24 \times 2 + 0.14 \times 0 + 0.06 \times 1 = 1.1$
t_3	$0.56 \times 2 + 0.24 \times 1 + 0.14 \times 1 + 0.06 \times 0 = 1.5$

Expected Ranks Example

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$

world W	$\Pr[W]$
$\{t_1 = 120, t_2 = 103, t_3 = 98\}$	$0.8 \times 0.7 \times 1 = 0.56$
$\{t_1 = 120, t_3 = 98, t_2 = 70\}$	$0.8 \times 0.3 \times 1 = 0.24$
$\{t_2 = 103, t_3 = 98, t_1 = 62\}$	$0.2 \times 0.7 \times 1 = 0.14$
$\{t_3 = 98, t_2 = 70, t_1 = 62\}$	$0.2 \times 0.3 \times 1 = 0.06$

tuple	r(tuple)
t_1	$0.56 \times 0 + 0.24 \times 0 + 0.14 \times 2 + 0.06 \times 2 = 0.4$
t_2	$0.56 \times 1 + 0.24 \times 2 + 0.14 \times 0 + 0.06 \times 1 = 1.1$
t_3	$0.56 \times 2 + 0.24 \times 1 + 0.14 \times 1 + 0.06 \times 0 = 1.5$

Expected Ranks Example

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$

world W	$\Pr[W]$
$\{t_1 = 120, t_2 = 103, t_3 = 98\}$	$0.8 \times 0.7 \times 1 = 0.56$
$\{t_1 = 120, t_3 = 98, t_2 = 70\}$	$0.8 \times 0.3 \times 1 = 0.24$
$\{t_2 = 103, t_3 = 98, t_1 = 62\}$	$0.2 \times 0.7 \times 1 = 0.14$
$\{t_3 = 98, t_2 = 70, t_1 = 62\}$	$0.2 \times 0.3 \times 1 = 0.06$

tuple	r(tuple)
t_1	$0.56 \times 0 + 0.24 \times 0 + 0.14 \times 2 + 0.06 \times 2 = 0.4$
t_2	$0.56 \times 1 + 0.24 \times 2 + 0.14 \times 0 + 0.06 \times 1 = 1.1$
t_3	$0.56 \times 2 + 0.24 \times 1 + 0.14 \times 1 + 0.06 \times 0 = 1.5$

Expected Ranks Example

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$

world W	$\Pr[W]$
$\{t_1 = 120, t_2 = 103, t_3 = 98\}$	$0.8 \times 0.7 \times 1 = 0.56$
$\{t_1 = 120, t_3 = 98, t_2 = 70\}$	$0.8 \times 0.3 \times 1 = 0.24$
$\{t_2 = 103, t_3 = 98, t_1 = 62\}$	$0.2 \times 0.7 \times 1 = 0.14$
$\{t_3 = 98, t_2 = 70, t_1 = 62\}$	$0.2 \times 0.3 \times 1 = 0.06$

tuple	r(tuple)
t_1	$0.56 \times 0 + 0.24 \times 0 + 0.14 \times 2 + 0.06 \times 2 = 0.4$
t_2	$0.56 \times 1 + 0.24 \times 2 + 0.14 \times 0 + 0.06 \times 1 = 1.1$
t_3	$0.56 \times 2 + 0.24 \times 1 + 0.14 \times 1 + 0.06 \times 0 = 1.5$

Expected Ranks Example

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$

world W	$\Pr[W]$
$\{t_1 = 120, t_2 = 103, t_3 = 98\}$	$0.8 \times 0.7 \times 1 = 0.56$
$\{t_1 = 120, t_3 = 98, t_2 = 70\}$	$0.8 \times 0.3 \times 1 = 0.24$
$\{t_2 = 103, t_3 = 98, t_1 = 62\}$	$0.2 \times 0.7 \times 1 = 0.14$
$\{t_3 = 98, t_2 = 70, t_1 = 62\}$	$0.2 \times 0.3 \times 1 = 0.06$

tuple	r(tuple)
t_1	$0.56 \times 0 + 0.24 \times 0 + 0.14 \times 2 + 0.06 \times 2 = 0.4$
t_2	$0.56 \times 1 + 0.24 \times 2 + 0.14 \times 0 + 0.06 \times 1 = 1.1$
t_3	$0.56 \times 2 + 0.24 \times 1 + 0.14 \times 1 + 0.06 \times 0 = 1.5$

Expected Ranks Example

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$

world W	$\Pr[W]$
$\{t_1 = 120, t_2 = 103, t_3 = 98\}$	$0.8 \times 0.7 \times 1 = 0.56$
$\{t_1 = 120, t_3 = 98, t_2 = 70\}$	$0.8 \times 0.3 \times 1 = 0.24$
$\{t_2 = 103, t_3 = 98, t_1 = 62\}$	$0.2 \times 0.7 \times 1 = 0.14$
$\{t_3 = 98, t_2 = 70, t_1 = 62\}$	$0.2 \times 0.3 \times 1 = 0.06$

tuple	r(tuple)
t_1	$0.56 \times 0 + 0.24 \times 0 + 0.14 \times 2 + 0.06 \times 2 = 0.4$
t_2	$0.56 \times 1 + 0.24 \times 2 + 0.14 \times 0 + 0.06 \times 1 = 1.1$
t_3	$0.56 \times 2 + 0.24 \times 1 + 0.14 \times 1 + 0.06 \times 0 = 1.5$

Expected Ranks Example

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$

world W	$\Pr[W]$
$\{t_1 = 120, t_2 = 103, t_3 = 98\}$	$0.8 \times 0.7 \times 1 = 0.56$
$\{t_1 = 120, t_3 = 98, t_2 = 70\}$	$0.8 \times 0.3 \times 1 = 0.24$
$\{t_2 = 103, t_3 = 98, t_1 = 62\}$	$0.2 \times 0.7 \times 1 = 0.14$
$\{t_3 = 98, t_2 = 70, t_1 = 62\}$	$0.2 \times 0.3 \times 1 = 0.06$

tuple	r(tuple)
t_1	$0.56 \times 0 + 0.24 \times 0 + 0.14 \times 2 + 0.06 \times 2 = 0.4$
t_2	$0.56 \times 1 + 0.24 \times 2 + 0.14 \times 0 + 0.06 \times 1 = 1.1$
t_3	$0.56 \times 2 + 0.24 \times 1 + 0.14 \times 1 + 0.06 \times 0 = 1.5$

Expected Ranks Example

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$

world W	$\Pr[W]$
$\{t_1 = 120, t_2 = 103, t_3 = 98\}$	$0.8 \times 0.7 \times 1 = 0.56$
$\{t_1 = 120, t_3 = 98, t_2 = 70\}$	$0.8 \times 0.3 \times 1 = 0.24$
$\{t_2 = 103, t_3 = 98, t_1 = 62\}$	$0.2 \times 0.7 \times 1 = 0.14$
$\{t_3 = 98, t_2 = 70, t_1 = 62\}$	$0.2 \times 0.3 \times 1 = 0.06$

tuple	r(tuple)
t_1	$0.56 \times 0 + 0.24 \times 0 + 0.14 \times 2 + 0.06 \times 2 = 0.4$
t_2	$0.56 \times 1 + 0.24 \times 2 + 0.14 \times 0 + 0.06 \times 1 = 1.1$
t_3	$0.56 \times 2 + 0.24 \times 1 + 0.14 \times 1 + 0.06 \times 0 = 1.5$

Expected Ranks Example

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$

world W	$\Pr[W]$
$\{t_1 = 120, t_2 = 103, t_3 = 98\}$	$0.8 \times 0.7 \times 1 = 0.56$
$\{t_1 = 120, t_3 = 98, t_2 = 70\}$	$0.8 \times 0.3 \times 1 = 0.24$
$\{t_2 = 103, t_3 = 98, t_1 = 62\}$	$0.2 \times 0.7 \times 1 = 0.14$
$\{t_3 = 98, t_2 = 70, t_1 = 62\}$	$0.2 \times 0.3 \times 1 = 0.06$

tuple	r(tuple)
t_1	$0.56 \times 0 + 0.24 \times 0 + 0.14 \times 2 + 0.06 \times 2 = 0.4$
t_2	$0.56 \times 1 + 0.24 \times 2 + 0.14 \times 0 + 0.06 \times 1 = 1.1$
t_3	$0.56 \times 2 + 0.24 \times 1 + 0.14 \times 1 + 0.06 \times 0 = 1.5$

Expected Ranks Example

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$

world W	$\Pr[W]$
$\{t_1 = 120, t_2 = 103, t_3 = 98\}$	$0.8 \times 0.7 \times 1 = 0.56$
$\{t_1 = 120, t_3 = 98, t_2 = 70\}$	$0.8 \times 0.3 \times 1 = 0.24$
$\{t_2 = 103, t_3 = 98, t_1 = 62\}$	$0.2 \times 0.7 \times 1 = 0.14$
$\{t_3 = 98, t_2 = 70, t_1 = 62\}$	$0.2 \times 0.3 \times 1 = 0.06$

tuple	r(tuple)
t_1	$0.56 \times 0 + 0.24 \times 0 + 0.14 \times 2 + 0.06 \times 2 = 0.4$
t_2	$0.56 \times 1 + 0.24 \times 2 + 0.14 \times 0 + 0.06 \times 1 = 1.1$
t_3	$0.56 \times 2 + 0.24 \times 1 + 0.14 \times 1 + 0.06 \times 0 = 1.5$

Expected Ranks Example

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$

world W	$\Pr[W]$
$\{t_1 = 120, t_2 = 103, t_3 = 98\}$	$0.8 \times 0.7 \times 1 = 0.56$
$\{t_1 = 120, t_3 = 98, t_2 = 70\}$	$0.8 \times 0.3 \times 1 = 0.24$
$\{t_2 = 103, t_3 = 98, t_1 = 62\}$	$0.2 \times 0.7 \times 1 = 0.14$
$\{t_3 = 98, t_2 = 70, t_1 = 62\}$	$0.2 \times 0.3 \times 1 = 0.06$

tuple	r(tuple)
t_1	$0.56 \times 0 + 0.24 \times 0 + 0.14 \times 2 + 0.06 \times 2 = 0.4$
t_2	$0.56 \times 1 + 0.24 \times 2 + 0.14 \times 0 + 0.06 \times 1 = 1.1$
t_3	$0.56 \times 2 + 0.24 \times 1 + 0.14 \times 1 + 0.06 \times 0 = 1.5$

Expected Ranks

- It has been shown that $r(t_i)$ may be written as

$$r(t_i) = \sum_{l=1}^{b_i} p_{i,l}(q(v_{i,l}) - Pr[X_i > v_{i,l}]) \qquad (2)$$

where,

b_i	$=$	number of choices in the pdf of t_i
$p_{i,l}$	$=$	probability of choice l in tuple t_i
$q(v_{i,l})$	$=$	$\sum_j Pr[X_j > v_{i,l}]$
X_i	$=$	pdf of tuple t_i
$Pr[X_i > v_{i,l}]$	$=$	contribution of t_i to $q(v_{i,l})$

Expected Ranks

- It has been shown that $r(t_i)$ may be written as

$$r(t_i) = \sum_{l=1}^{b_i} p_{i,l}(q(v_{i,l}) - Pr[X_i > v_{i,l}]) \quad (2)$$

where,

b_i	$=$	number of choices in the pdf of t_i
$p_{i,l}$	$=$	probability of choice l in tuple t_i
$q(v_{i,l})$	$=$	$\sum_j Pr[X_j > v_{i,l}]$
X_i	$=$	pdf of tuple t_i
$Pr[X_i > v_{i,l}]$	$=$	contribution of t_i to $q(v_{i,l})$

- $q(v_{i,l})$ is the sum of the probabilities that a tuple will outrank a tuple with score $v_{i,l}$

Expected Ranks

- It has been shown that $r(t_i)$ may be written as

$$r(t_i) = \sum_{l=1}^{b_i} p_{i,l}(q(v_{i,l}) - Pr[X_i > v_{i,l}]) \quad (2)$$

where,

b_i	$=$	number of choices in the pdf of t_i
$p_{i,l}$	$=$	probability of choice l in tuple t_i
$q(v_{i,l})$	$=$	$\sum_j Pr[X_j > v_{i,l}]$
X_i	$=$	pdf of tuple t_i
$Pr[X_i > v_{i,l}]$	$=$	contribution of t_i to $q(v_{i,l})$

- X_i may contain value-probability pairs (v, p) s.t. $v > v_{i,l}$, since the existence of $t_i = v_{i,l}$ precludes $t_i = v$, we must subtract the corresponding p's from $q(v_{i,l})$

Expected Ranks

- It has been shown that $r(t_i)$ may be written as

$$r(t_i) = \sum_{l=1}^{b_i} p_{i,l}(q(v_{i,l}) - Pr[X_i > v_{i,l}]) \quad (2)$$

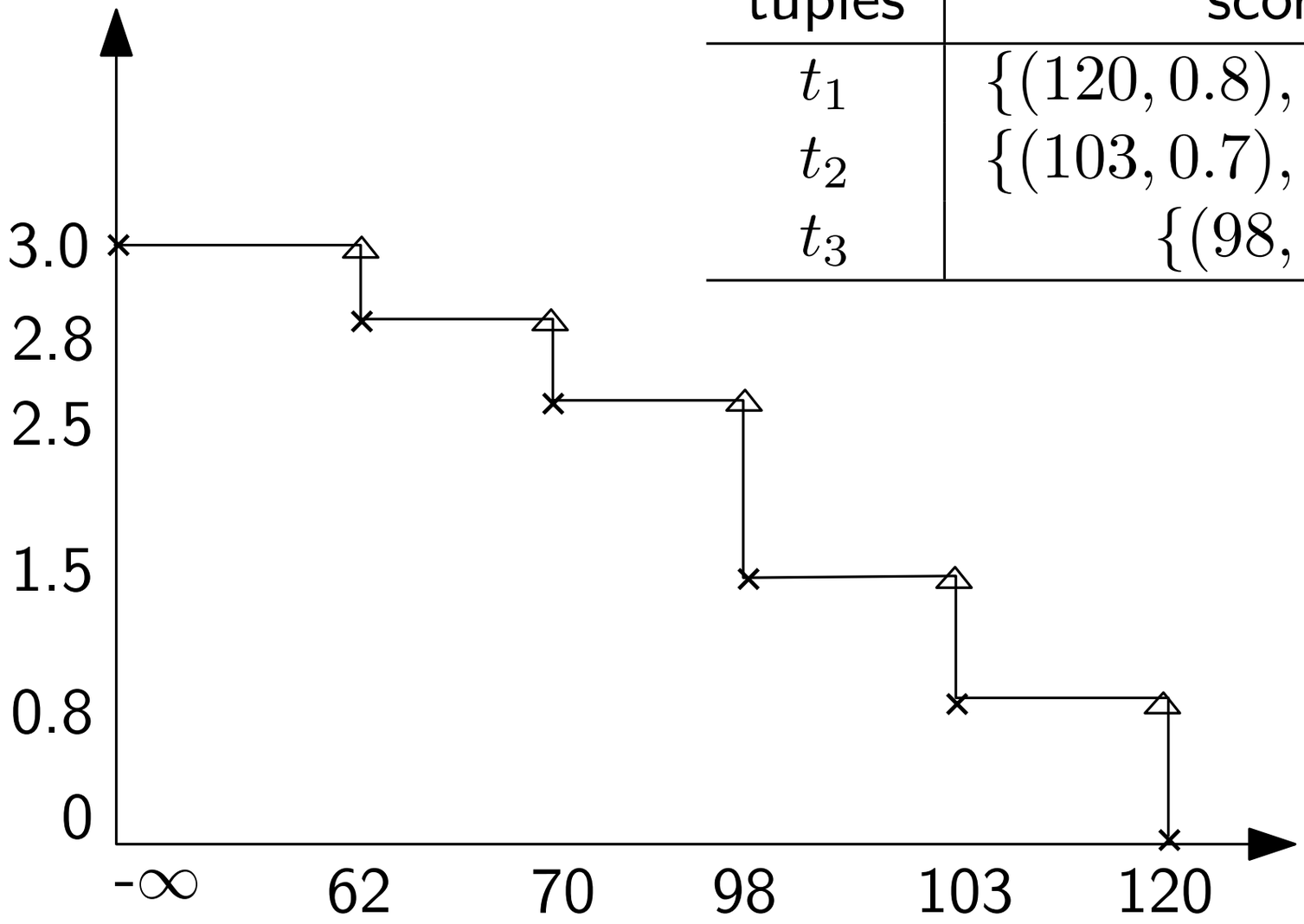
where,

b_i	$=$	number of choices in the pdf of t_i
$p_{i,l}$	$=$	probability of choice l in tuple t_i
$q(v_{i,l})$	$=$	$\sum_j Pr[X_j > v_{i,l}]$
X_i	$=$	pdf of tuple t_i
$Pr[X_i > v_{i,l}]$	$=$	contribution of t_i to $q(v_{i,l})$

- Efficient algorithms exist to compute the Expected ranks in $O(NlogN)$ time for a database of N tuples

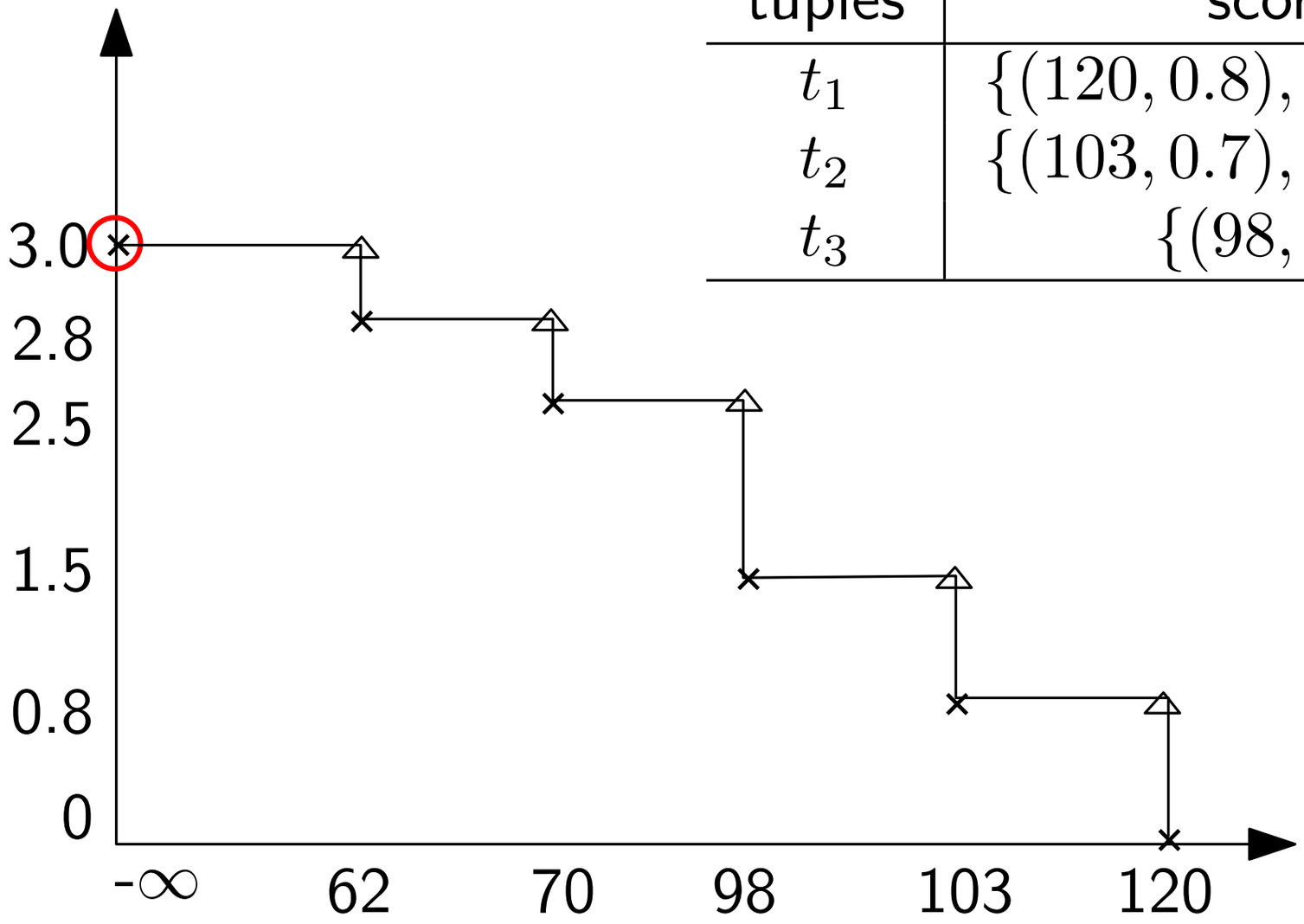
Computing Expected Ranks by $q(v)$'s

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$



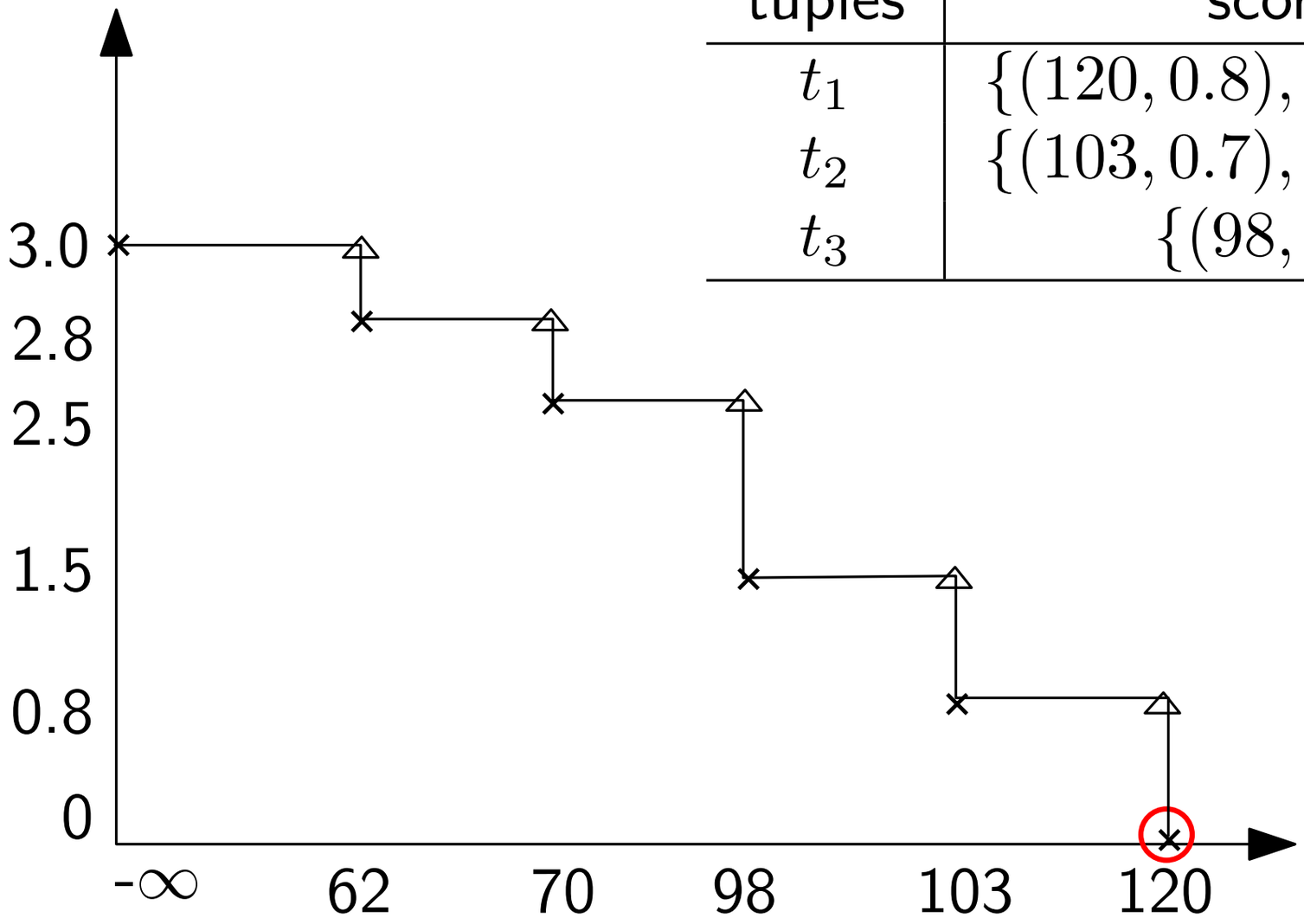
Computing Expected Ranks by $q(v)$'s

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$



Computing Expected Ranks by $q(v)$'s

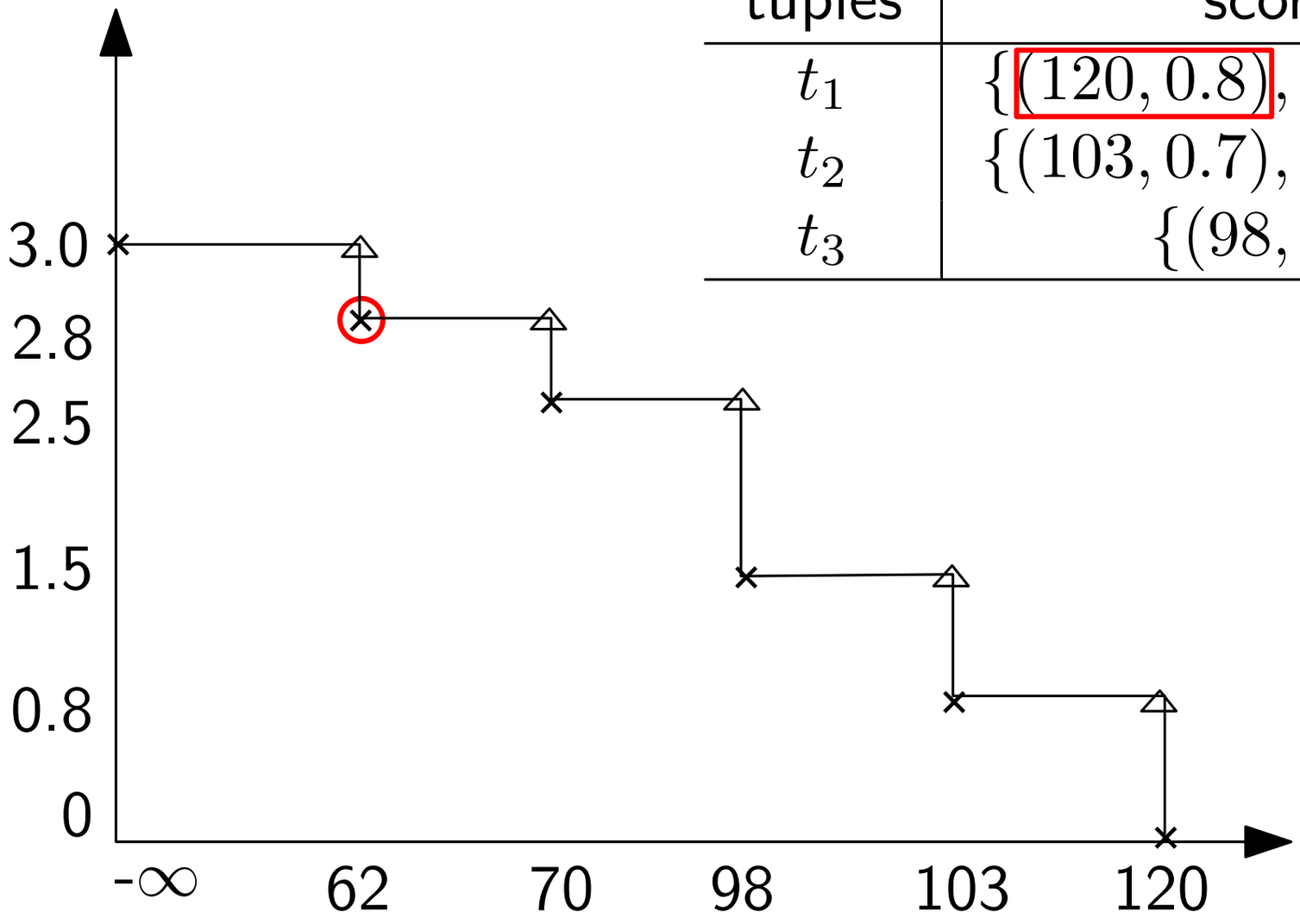
tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$



- $r(t_1) = 0.8 \times 0$

Computing Expected Ranks by $q(v)$'s

tuples	score
t_1	$\{(120, 0.8), (62, 0.2)\}$
t_2	$\{(103, 0.7), (70, 0.3)\}$
t_3	$\{(98, 1)\}$



- $r(t_1) = 0.8 \times 0 + 0.2 \times (2.8 - 0.8) = 0.4$

Distributed Probabilistic Data Model

site 1	
tuples	score
$t_{1,1}$	$X_{1,1}$
$t_{1,2}$	$X_{1,2}$
$\vdots$	$\vdots$

$\vdots$

site m	
tuples	score
$t_{2,1}$	$X_{2,1}$
$t_{2,2}$	$X_{2,2}$
$\vdots$	$\vdots$

Distributed Probabilistic Data Model

site 1	
tuples	score
$t_{1,1}$	$X_{1,1}$
$t_{1,2}$	$X_{1,2}$
$\vdots$	$\vdots$

$\vdots$

site m	
tuples	score
$t_{2,1}$	$X_{2,1}$
$t_{2,2}$	$X_{2,2}$
$\vdots$	$\vdots$

tuples
t_1
t_2
$\vdots$
t_N

Conceptual Database D

- We can think of the union of the individual databases D_i at each site s_i as a conceptual database D

Ranking Queries for Distributed Probabilistic Data

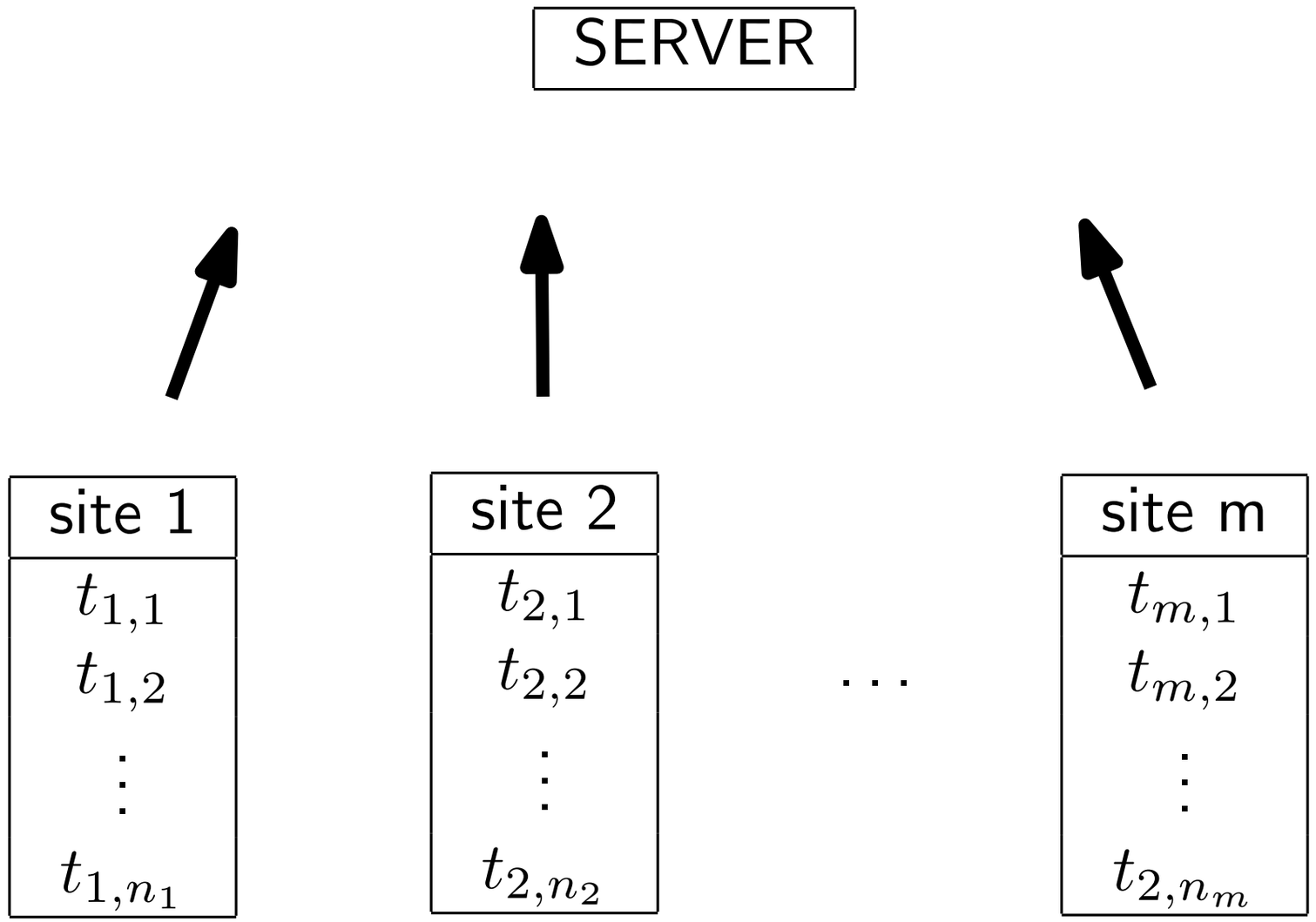
- We introduce two frameworks for ranking queries for distributed probabilistic data
  - Sorted Access on Local Ranks
  - Sorted Access on Expected Scores

Sorted Access on Local Ranks Framework

site 1	site 2		site m
$t_{1,1}$	$t_{2,1}$		$t_{m,1}$
$t_{1,2}$	$t_{2,2}$	$\cdots$	$t_{m,2}$
$\vdots$	$\vdots$		$\vdots$
t_{1,n_1}	t_{2,n_2}		t_{2,n_m}

- Every site calculates the **local ranks** of its tuples and stores tuples in ascending order of local ranks

Sorted Access on Local Ranks Framework



- The server accesses tuples in ascending order of local ranks and combines the **local ranks** to get the **global ranks**

Local and Global Ranks

- The **local rank** of a tuple $t_{i,j}$ at a site s_i in database D_i is

$$r(t_{i,j}, D_i) = \sum_{l=0}^{b_{i,j}} p_{i,j,l}(q_i(v_{i,j,l}) - Pr[X_{i,j} > v_{i,j,l}]) \quad (3)$$

- The **local rank** for a tuple $t_{i,j}$ at a site s_y with database D_y, s.t. $i \neq y$ is

$$r(t_{i,j}, D_y) = \sum_{l=1}^{b_{i,j}} p_{i,j,l}(q_y(v_{i,j,l})) \quad (4)$$

- The **global rank** for a tuple $t_{i,j}$ is

$$r(t_{i,j}, D_y) = \sum_{y=1}^{m} r(t_{i,j}, Dy) \quad (5)$$

Local and Global Ranks

- The **local rank** of a tuple $t_{i,j}$ at a site s_i in database D_i is

$$r(t_{i,j}, D_i) = \sum_{l=0}^{b_{i,j}} p_{i,j,l}(q_i(v_{i,j,l}) - Pr[X_{i,j} > v_{i,j,l}]) \quad (3)$$

- The **local rank** for a tuple $t_{i,j}$ at a site s_y with database D_y, s.t. $i \neq y$ is

$$r(t_{i,j}, D_y) = \sum_{l=1}^{b_{i,j}} p_{i,j,l}(q_y(v_{i,j,l})) \quad (4)$$

Local and Global Ranks

- The **local rank** of a tuple $t_{i,j}$ at a site s_i in database D_i is

$$r(t_{i,j}, D_i) = \sum_{l=0}^{b_{i,j}} p_{i,j,l}(q_i(v_{i,j,l}) - Pr[X_{i,j} > v_{i,j,l}]) \quad (3)$$

- The **local rank** for a tuple $t_{i,j}$ at a site s_y with database D_y, s.t. $i \neq y$ is

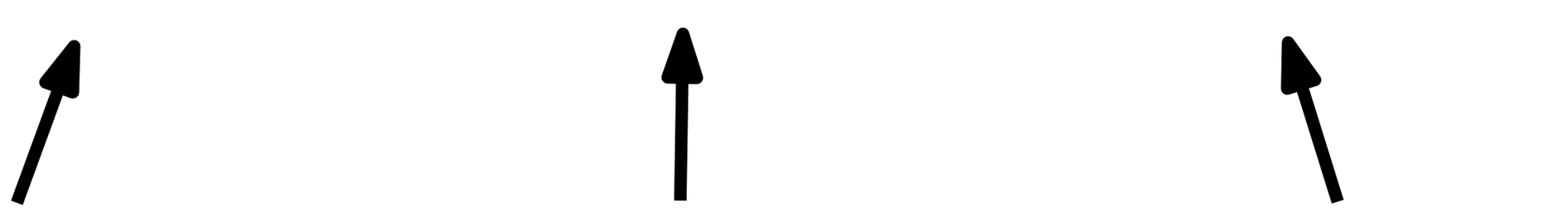
$$r(t_{i,j}, D_y) = \sum_{l=1}^{b_{i,j}} p_{i,j,l}(q_y(v_{i,j,l})) \quad (4)$$

- The **global rank** for a tuple $t_{i,j}$ is

$$r(t_{i,j}, D_y) = \sum_{y=1}^{m} r(t_{i,j}, Dy) \quad (5)$$

Sorted Access on Local Ranks Initialization

Rep. Queue	
tuple	lrank
$t_{3,1}$	0.8
$t_{1,1}$	1.2
$t_{2,1}$	2.3



site 1		
	tuple	lrank
$\rightarrow$	$t_{1,1}$	1.2
$\rightarrow$	$t_{1,2}$	5.9
	$\vdots$	
	t_{1,n_1}	34.2

site 2		
	tuple	lrank
$\rightarrow$	$t_{2,1}$	2.3
$\rightarrow$	$t_{2,2}$	3.4
	$\vdots$	
	t_{2,n_2}	29.1

site 3		
	tuple	lrank
$\rightarrow$	$t_{3,1}$	0.8
$\rightarrow$	$t_{3,2}$	4.1
	$\vdots$	
	t_{3,n_3}	40.4

Sorted Access on Local Ranks Initialization

Rep. Queue	
tuple	lrank
$t_{3,1}$	0.8
$t_{1,1}$	1.2
$t_{2,1}$	2.3

site 1		
	tuple	lrank
	$t_{1,1}$	1.2
$\rightarrow$	$t_{1,2}$	5.9
	$\vdots$	
	t_{1,n_1}	34.2

site 2		
	tuple	lrank
	$t_{2,1}$	2.3
$\rightarrow$	$t_{2,2}$	3.4
	$\vdots$	
	t_{2,n_2}	29.1

site 3		
	tuple	lrank
	$t_{3,1}$	0.8
$\rightarrow$	$t_{3,2}$	4.1
	$\vdots$	
	t_{3,n_3}	40.4

Sorted Access on Local Ranks: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{1,1}$	7.9

Rep. Queue	
tuple	lrank
$t_{2,2}$	3.4
$t_{3,2}$	4.1
$t_{1,2}$	5.9

site 1		
	tuple	lrank
	$t_{1,1}$	1.2
	$t_{1,2}$	5.9
$\rightarrow$	$\vdots$	
	t_{1,n_1}	34.2

site 2		
	tuple	lrank
	$t_{2,1}$	2.3
	$t_{2,2}$	3.4
$\rightarrow$	$\vdots$	
	t_{2,n_2}	29.1

site 3		
	tuple	lrank
	$t_{3,1}$	0.8
	$t_{3,2}$	4.1
$\rightarrow$	$\vdots$	
	t_{3,n_3}	40.4

Sorted Access on Local Ranks: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{1,1}$	7.9

tuple	lrank
$t_{2,2}$	3.4

←

Rep. Queue	
tuple	lrank
~~$t_{2,2}$~~	~~3.4~~
$t_{3,2}$	4.1
$t_{1,2}$	5.9

site 1		
	tuple	lrank
	$t_{1,1}$	1.2
	$t_{1,2}$	5.9
$\rightarrow$	$\vdots$	
	t_{1,n_1}	34.2

site 2		
	tuple	lrank
	$t_{2,1}$	2.3
	$t_{2,2}$	3.4
$\rightarrow$	$\vdots$	
	t_{2,n_2}	29.1

site 3		
	tuple	lrank
	$t_{3,1}$	0.8
	$t_{3,2}$	4.1
$\rightarrow$	$\vdots$	
	t_{3,n_3}	40.4

Sorted Access on Local Ranks: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{1,1}$	7.9

tuple	lrank
$t_{2,2}$	3.4

Rep. Queue	
tuple	lrank
~~$t_{2,2}$~~	~~3.4~~
$t_{3,2}$	4.1
$t_{1,2}$	5.9

tuple	lrank
$t_{2,3}$	4.8

site 1		
	tuple	lrank
	$t_{1,1}$	1.2
	$t_{1,2}$	5.9
$\rightarrow$	$\vdots$	
	t_{1,n_1}	34.2

site 2		
	tuple	lrank
	$t_{2,1}$	2.3
	$t_{2,2}$	3.4
$\rightarrow$	$\vdots$	
	t_{2,n_2}	29.1

site 3		
	tuple	lrank
	$t_{3,1}$	0.8
	$t_{3,2}$	4.1
$\rightarrow$	$\vdots$	
	t_{3,n_3}	40.4

Sorted Access on Local Ranks: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{1,1}$	7.9

tuple	lrank
$t_{2,2}$	3.4

Rep. Queue	
tuple	lrank
$t_{3,2}$	4.1
$t_{2,3}$	4.8
$t_{1,2}$	5.9

site 1		
	tuple	lrank
	$t_{1,1}$	1.2
	$t_{1,2}$	5.9
$\rightarrow$	$\vdots$	
	t_{1,n_1}	34.2

site 2		
	tuple	lrank
	$t_{2,1}$	2.3
	$t_{2,2}$	3.4
$\rightarrow$	$\vdots$	
	t_{2,n_2}	29.1

site 3		
	tuple	lrank
	$t_{3,1}$	0.8
	$t_{3,2}$	4.1
$\rightarrow$	$\vdots$	
	t_{3,n_3}	40.4

Sorted Access on Local Ranks: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{1,1}$	7.9

tuple	lrank
$t_{2,2}$	3.4

Rep. Queue	
tuple	lrank
$t_{3,2}$	4.1
$t_{2,3}$	4.8
$t_{1,2}$	5.9

$X_{2,2}$

$X_{2,2}$

site 1		
	tuple	lrank
	$t_{1,1}$	1.2
	$t_{1,2}$	5.9
$\rightarrow$	$\vdots$	
	t_{1,n_1}	34.2

site 2		
	tuple	lrank
	$t_{2,1}$	2.3
	$t_{2,2}$	3.4
$\rightarrow$	$\vdots$	
	t_{2,n_2}	29.1

site 3		
	tuple	lrank
	$t_{3,1}$	0.8
	$t_{3,2}$	4.1
$\rightarrow$	$\vdots$	
	t_{3,n_3}	40.4

Sorted Access on Local Ranks: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{1,1}$	7.9

tuple	lrank
$t_{2,2}$	3.4

Rep. Queue	
tuple	lrank
$t_{3,2}$	4.1
$t_{2,3}$	4.8
$t_{1,2}$	5.9

lrank
1.5

lrank
0.7

site 1		
	tuple	lrank
	$t_{1,1}$	1.2
	$t_{1,2}$	5.9
$\rightarrow$	$\vdots$	
	t_{1,n_1}	34.2

site 2		
	tuple	lrank
	$t_{2,1}$	2.3
	$t_{2,2}$	3.4
$\rightarrow$	$\vdots$	
	t_{2,n_2}	29.1

site 3		
	tuple	lrank
	$t_{3,1}$	0.8
	$t_{3,2}$	4.1
$\rightarrow$	$\vdots$	
	t_{3,n_3}	40.4

Sorted Access on Local Ranks: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{1,1}$	7.9

tuple	lrank
$t_{2,2}$	3.4

grank
5.6

Rep. Queue	
tuple	lrank
$t_{3,2}$	4.1
$t_{2,3}$	4.8
$t_{1,2}$	5.9

lrank
1.5

lrank
0.7

site 1		
	tuple	lrank
	$t_{1,1}$	1.2
	$t_{1,2}$	5.9
$\rightarrow$	$\vdots$	
	t_{1,n_1}	34.2

site 2		
	tuple	lrank
	$t_{2,1}$	2.3
	$t_{2,2}$	3.4
$\rightarrow$	$\vdots$	
	t_{2,n_2}	29.1

site 3		
	tuple	lrank
	$t_{3,1}$	0.8
	$t_{3,2}$	4.1
$\rightarrow$	$\vdots$	
	t_{3,n_3}	40.4

Sorted Access on Local Ranks: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
~~$t_{1,1}$~~	~~7.9~~

tuple	grank
$t_{2,2}$	5.6

Rep. Queue	
tuple	lrank
$t_{3,2}$	4.1
$t_{2,3}$	4.8
$t_{1,2}$	5.9

site 1		
	tuple	lrank
	$t_{1,1}$	1.2
	$t_{1,2}$	5.9
$\rightarrow$	$\vdots$	
	t_{1,n_1}	34.2

site 2		
	tuple	lrank
	$t_{2,1}$	2.3
	$t_{2,2}$	3.4
$\rightarrow$	$\vdots$	
	t_{2,n_2}	29.1

site 3		
	tuple	lrank
	$t_{3,1}$	0.8
	$t_{3,2}$	4.1
$\rightarrow$	$\vdots$	
	t_{3,n_3}	40.4

Sorted Access on Local Ranks: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{2,2}$	5.6

Rep. Queue	
tuple	lrank
$t_{3,2}$	4.1
$t_{2,3}$	4.8
$t_{1,2}$	5.9

site 1		
	tuple	lrank
	$t_{1,1}$	1.2
	$t_{1,2}$	5.9
$\rightarrow$	$\vdots$	
	t_{1,n_1}	34.2

site 2		
	tuple	lrank
	$t_{2,1}$	2.3
	$t_{2,2}$	3.4
$\rightarrow$	$\vdots$	
	t_{2,n_2}	29.1

site 3		
	tuple	lrank
	$t_{3,1}$	0.8
	$t_{3,2}$	4.1
$\rightarrow$	$\vdots$	
	t_{3,n_3}	40.4

Sorted Access on Local Ranks: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{2,2}$	5.6

We can safely terminate whenever the largest grank from $top-k$ queue is $\leq$ smallest lrank from Rep. Queue

Rep. Queue	
tuple	lrank
$t_{3,2}$	4.1
$t_{2,3}$	4.8
$t_{1,2}$	5.9

site 1		
	tuple	lrank
	$t_{1,1}$	1.2
	$t_{1,2}$	5.9
$\rightarrow$	$\vdots$	
	t_{1,n_1}	34.2

site 2		
	tuple	lrank
	$t_{2,1}$	2.3
	$t_{2,2}$	3.4
$\rightarrow$	$\vdots$	
	t_{2,n_2}	29.1

site 3		
	tuple	lrank
	$t_{3,1}$	0.8
	$t_{3,2}$	4.1
$\rightarrow$	$\vdots$	
	t_{3,n_3}	40.4

Sorted Access on Local Ranks: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{2,2}$	5.6

We can safely terminate whenever the largest grank from $top-k$ queue is $\leq$ smallest lrank from Rep. Queue

Rep. Queue	
tuple	lrank
$t_{3,2}$	4.1
$t_{2,3}$	4.8
$t_{1,2}$	5.9

A-LR

site 1		
	tuple	lrank
	$t_{1,1}$	1.2
	$t_{1,2}$	5.9
$\rightarrow$	$\vdots$	
	t_{1,n_1}	34.2

site 2		
	tuple	lrank
	$t_{2,1}$	2.3
	$t_{2,2}$	3.4
$\rightarrow$	$\vdots$	
	t_{2,n_2}	29.1

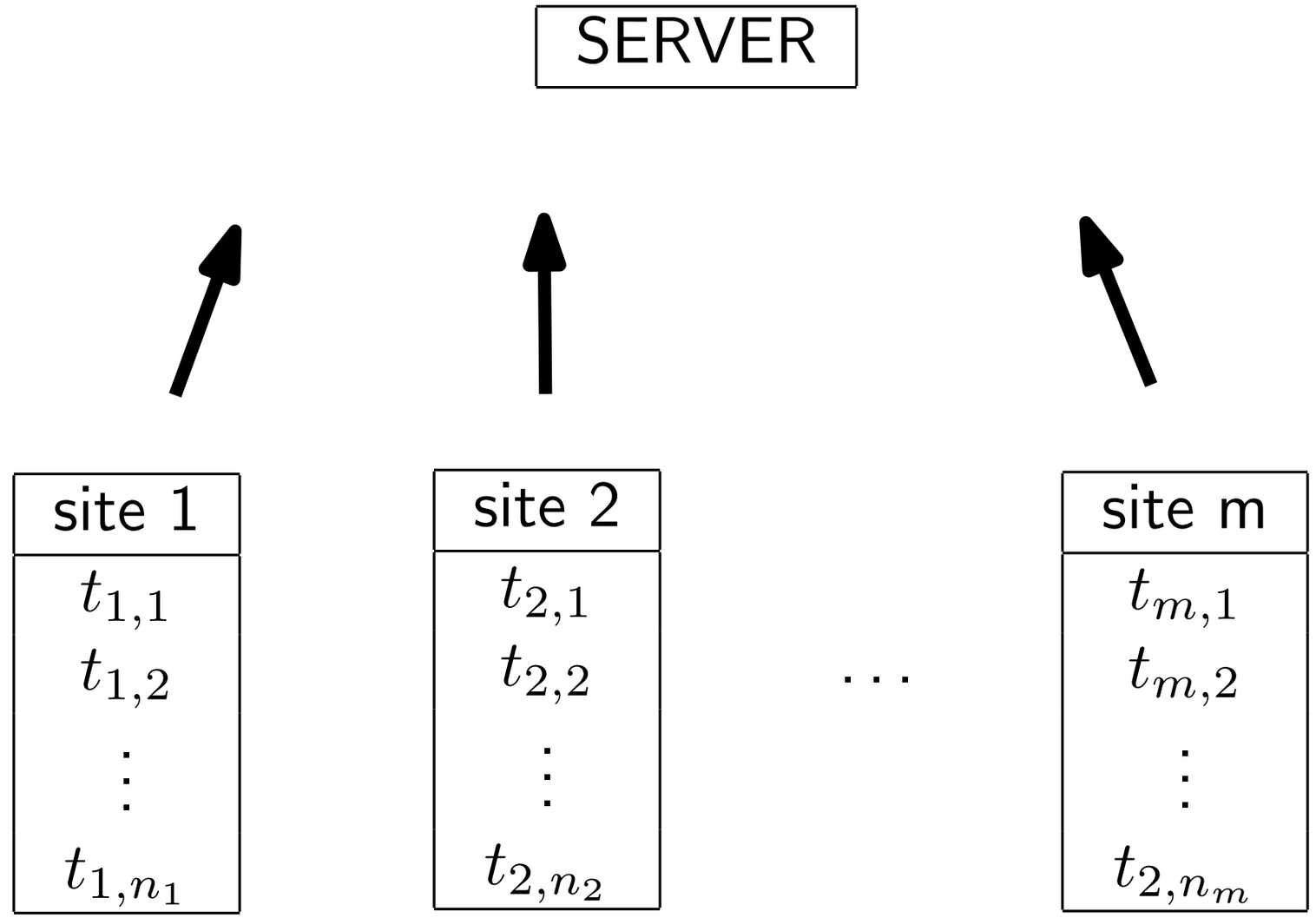
site 3		
	tuple	lrank
	$t_{3,1}$	0.8
	$t_{3,2}$	4.1
$\rightarrow$	$\vdots$	
	t_{3,n_3}	40.4

Sorted Access on Expected Scores Framework

site 1	site 2		site m
$t_{1,1}$	$t_{2,1}$		$t_{m,1}$
$t_{1,2}$	$t_{2,2}$	$\cdots$	$t_{m,2}$
$\vdots$	$\vdots$		$\vdots$
t_{1,n_1}	t_{2,n_2}		t_{2,n_m}

- Every site calculates the **local ranks** and the **expected scores** of its tuples and stores the tuples in descending order of expected scores

Sorted Access on Expected Scores Framework



- Tuples are accessed by descending order of expected scores and the server calculates **global ranks**

Sorted Access on Expected Scores Initialization

site 1		
	tuple	$E[X]$
$\rightarrow$	$t_{1,1}$	489
	$t_{1,2}$	421
	$\vdots$	
	t_{1,n_1}	5

site 2		
	tuple	$E[X]$
$\rightarrow$	$t_{2,1}$	476
	$t_{2,2}$	464
	$\vdots$	
	t_{2,n_2}	11

site 3		
	tuple	$E[X]$
$\rightarrow$	$t_{3,1}$	500
	$t_{3,2}$	432
	$\vdots$	
	t_{3,n_3}	1

Sorted Access on Expected Scores Initialization

Rep. Queue	
tuple	$E[X]$
$t_{3,1}$	500
$t_{1,1}$	489
$t_{2,1}$	476

site 1		
	tuple	$E[X]$
	$t_{1,1}$	489
$\rightarrow$	$t_{1,2}$	421
	$\vdots$	
	t_{1,n_1}	5

site 2		
	tuple	$E[X]$
	$t_{2,1}$	476
$\rightarrow$	$t_{2,2}$	464
	$\vdots$	
	t_{2,n_2}	11

site 3		
	tuple	$E[X]$
	$t_{3,1}$	500
$\rightarrow$	$t_{3,2}$	432
	$\vdots$	
	t_{3,n_3}	1

Sorted Access on Expected Scores: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{1,1}$	7.9

Rep. Queue	
tuple	$E[X]$
$t_{2,2}$	464
$t_{3,2}$	432
$t_{1,2}$	421

site 1		
	tuple	$E[X]$
	$t_{1,1}$	489
	$t_{1,2}$	421
$\rightarrow$	$\vdots$	
	t_{1,n_1}	5

site 2		
	tuple	$E[X]$
	$t_{2,1}$	476
	$t_{2,2}$	464
$\rightarrow$	$\vdots$	
	t_{2,n_2}	11

site 3		
	tuple	$E[X]$
	$t_{3,1}$	500
	$t_{3,2}$	432
$\rightarrow$	$\vdots$	
	t_{3,n_3}	1

Sorted Access on Expected Scores: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{1,1}$	7.9

tuple	lrank
$t_{2,2}$	3.4

Rep. Queue	
tuple	$E[X]$
~~$t_{2,2}$~~	~~464~~
$t_{3,2}$	432
$t_{1,2}$	421

site 1		
	tuple	$E[X]$
	$t_{1,1}$	489
	$t_{1,2}$	421
$\rightarrow$	$\vdots$	
	t_{1,n_1}	5

site 2		
	tuple	$E[X]$
	$t_{2,1}$	476
	$t_{2,2}$	464
$\rightarrow$	$\vdots$	
	t_{2,n_2}	11

site 3		
	tuple	$E[X]$
	$t_{3,1}$	500
	$t_{3,2}$	432
$\rightarrow$	$\vdots$	
	t_{3,n_3}	1

Sorted Access on Expected Scores: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{1,1}$	7.9

tuple	lrank
$t_{2,2}$	3.4

Rep. Queue	
tuple	$E[X]$
~~$t_{2,2}$~~	~~464~~
$t_{3,2}$	432
$t_{1,2}$	421

tuple	$E[X]$
$t_{2,3}$	429

site 1		
	tuple	$E[X]$
	$t_{1,1}$	489
	$t_{1,2}$	421
$\rightarrow$	$\vdots$	
	t_{1,n_1}	5

site 2		
	tuple	$E[X]$
	$t_{2,1}$	476
	$t_{2,2}$	464
$\rightarrow$	$\vdots$	
	t_{2,n_2}	11

site 3		
	tuple	$E[X]$
	$t_{3,1}$	500
	$t_{3,2}$	432
$\rightarrow$	$\vdots$	
	t_{3,n_3}	1

Sorted Access on Expected Scores: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{1,1}$	7.9

tuple	lrank
$t_{2,2}$	3.4

Rep. Queue	
tuple	$E[X]$
$t_{3,2}$	432
$t_{2,3}$	429
$t_{1,2}$	421

site 1		
	tuple	$E[X]$
	$t_{1,1}$	489
	$t_{1,2}$	421
$\rightarrow$	$\vdots$	
	t_{1,n_1}	5

site 2		
	tuple	$E[X]$
	$t_{2,1}$	476
	$t_{2,2}$	464
$\rightarrow$	$\vdots$	
	t_{2,n_2}	11

site 3		
	tuple	$E[X]$
	$t_{3,1}$	500
	$t_{3,2}$	432
$\rightarrow$	$\vdots$	
	t_{3,n_3}	1

Sorted Access on Expected Scores: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{1,1}$	7.9

tuple	lrank
$t_{2,2}$	3.4

Rep. Queue	
tuple	$E[X]$
$t_{3,2}$	432
$t_{2,3}$	429
$t_{1,2}$	421

$X_{2,2}$

$X_{2,2}$

site 1		
	tuple	$E[X]$
	$t_{1,1}$	489
	$t_{1,2}$	421
$\rightarrow$	$\vdots$	
	t_{1,n_1}	5

site 2		
	tuple	$E[X]$
	$t_{2,1}$	476
	$t_{2,2}$	464
$\rightarrow$	$\vdots$	
	t_{2,n_2}	11

site 3		
	tuple	$E[X]$
	$t_{3,1}$	500
	$t_{3,2}$	432
$\rightarrow$	$\vdots$	
	t_{3,n_3}	1

Sorted Access on Expected Scores: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{1,1}$	7.9

tuple	lrank
$t_{2,2}$	3.4

Rep. Queue	
tuple	$E[X]$
$t_{3,2}$	432
$t_{2,3}$	429
$t_{1,2}$	421

lrank
1.5

lrank
0.7

site 1		
	tuple	$E[X]$
	$t_{1,1}$	489
	$t_{1,2}$	421
$\rightarrow$	$\vdots$	
	t_{1,n_1}	5

site 2		
	tuple	$E[X]$
	$t_{2,1}$	476
	$t_{2,2}$	464
$\rightarrow$	$\vdots$	
	t_{2,n_2}	11

site 3		
	tuple	$E[X]$
	$t_{3,1}$	500
	$t_{3,2}$	432
$\rightarrow$	$\vdots$	
	t_{3,n_3}	1

Sorted Access on Expected Scores: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{1,1}$	7.9

tuple	lrank
$t_{2,2}$	3.4

grank
5.6

Rep. Queue	
tuple	$E[X]$
$t_{3,2}$	432
$t_{2,3}$	429
$t_{1,2}$	421

lrank
1.5

lrank
0.7

site 1		
	tuple	$E[X]$
	$t_{1,1}$	489
	$t_{1,2}$	421
$\rightarrow$	$\vdots$	
	t_{1,n_1}	5

site 2		
	tuple	$E[X]$
	$t_{2,1}$	476
	$t_{2,2}$	464
$\rightarrow$	$\vdots$	
	t_{2,n_2}	11

site 3		
	tuple	$E[X]$
	$t_{3,1}$	500
	$t_{3,2}$	432
$\rightarrow$	$\vdots$	
	t_{3,n_3}	1

Sorted Access on Expected Scores: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
~~$t_{1,1}$~~	~~7.9~~

tuple	grank
$t_{2,2}$	5.6

Rep. Queue	
tuple	$E[X]$
$t_{3,2}$	432
$t_{2,3}$	429
$t_{1,2}$	421

site 1		
	tuple	$E[X]$
	$t_{1,1}$	489
	$t_{1,2}$	421
$\rightarrow$	$\vdots$	
	t_{1,n_1}	5

site 2		
	tuple	$E[X]$
	$t_{2,1}$	476
	$t_{2,2}$	464
$\rightarrow$	$\vdots$	
	t_{2,n_2}	11

site 3		
	tuple	$E[X]$
	$t_{3,1}$	500
	$t_{3,2}$	432
$\rightarrow$	$\vdots$	
	t_{3,n_3}	1

Sorted Access on Expected Scores: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{2,2}$	5.6

Rep. Queue	
tuple	$E[X]$
$t_{3,2}$	432
$t_{2,3}$	429
$t_{1,2}$	421

site 1		
	tuple	$E[X]$
	$t_{1,1}$	489
	$t_{1,2}$	421
$\rightarrow$	$\vdots$	
	t_{1,n_1}	5

site 2		
	tuple	$E[X]$
	$t_{2,1}$	476
	$t_{2,2}$	464
$\rightarrow$	$\vdots$	
	t_{2,n_2}	11

site 3		
	tuple	$E[X]$
	$t_{3,1}$	500
	$t_{3,2}$	432
$\rightarrow$	$\vdots$	
	t_{3,n_3}	1

Sorted Access on Expected Scores: a Round

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{2,2}$	5.6

Now the only question is when may we safely terminate and be certain we have the global $top-k$

Rep. Queue	
tuple	$E[X]$
$t_{3,2}$	432
$t_{2,3}$	429
$t_{1,2}$	421

site 1		
	tuple	$E[X]$
	$t_{1,1}$	489
	$t_{1,2}$	421
$\rightarrow$	$\vdots$	
	t_{1,n_1}	5

site 2		
	tuple	$E[X]$
	$t_{2,1}$	476
	$t_{2,2}$	464
$\rightarrow$	$\vdots$	
	t_{2,n_2}	11

site 3		
	tuple	$E[X]$
	$t_{3,1}$	500
	$t_{3,2}$	432
$\rightarrow$	$\vdots$	
	t_{3,n_3}	1

Sorted Access on Expected Scores: Termination

- The largest element from the $top-k$ queue is clearly an upper bound r_λ^+ for the **global rank** of any seen tuple t with pdf X to be in the $top-k$ at round λ

Sorted Access on Expected Scores: Termination

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{2,2}$	5.6

Rep. Queue	
tuple	$E[X]$
$t_{3,2}$	432
$t_{2,3}$	429
$t_{1,2}$	421

- The largest element from the $top-k$ queue is clearly an upper bound r_λ^+ for the **global rank** of any seen tuple t with pdf X to be in the $top-k$ at round λ

Sorted Access on Expected Scores: Termination

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{2,2}$	5.6

Rep. Queue	
tuple	$E[X]$
$t_{3,2}$	432
$t_{2,3}$	429
$t_{1,2}$	421

- The largest element from the $top-k$ queue is clearly an upper bound r_λ^+ for the **global rank** of any seen tuple t with pdf X to be in the $top-k$ at round λ
- The head from the Representative queue with expectance τ is an upper bound for the expectance of any unseen t s.t. $E[X] \leq \tau$

Sorted Access on Expected Scores: Termination

$top-2$ Queue	
tuple	grank
$t_{2,1}$	5.4
$t_{2,2}$	5.6

Rep. Queue	
tuple	$E[X]$
$t_{3,2}$	432
$t_{2,3}$	429
$t_{1,2}$	421

- The largest element from the $top-k$ queue is clearly an upper bound r_λ^+ for the **global rank** of any seen tuple t with pdf X to be in the $top-k$ at round λ
- The head from the Representative queue with expectance τ is an upper bound for the expectance of any unseen t s.t. $E[X] \leq \tau$
- How can we derive a lower bound r_λ^- for the **global rank** of any unseen tuple t s.t. when $r_\lambda^+ \leq r_\lambda^-$ it is safe to terminate at round λ?

Sorted Access on Expected Scores: a Lower Bound?

- We introduce two methods to find a lower bound r_λ^- for any unseen tuple t at round λ

Sorted Access on Expected Scores: a Lower Bound?

- We introduce two methods to find a lower bound r_λ^- for any unseen tuple t at round λ
  - Markov Inequality

Sorted Access on Expected Scores: a Lower Bound?

- We introduce two methods to find a lower bound r_λ^- for any unseen tuple t at round λ
  - Markov Inequality
  - Linear Programming

Markov Inequality Lower Bound

- We know that the pdf of any unseen t must satisfy $E[X] \leq \tau$

Markov Inequality Lower Bound

- We know that the pdf of any unseen t must satisfy $E[X] \leq \tau$
- We can use the **Markov Inequality** to lower bound the rank of any site s_i with database D_i as,

$$r(t, D_i) = \sum_{j=1}^{n_i} \Pr[X_j > X] = n_i - \sum_{j=1}^{n_i} \Pr[X \geq X_j]$$

$$\geq n_i - \sum_{j=1}^{n_i} \sum_{\ell=1}^{b_{ij}} p_{i,j,\ell} \frac{E[X]}{v_{i,j,\ell}}. \qquad \text{(Markov Ineq.)}$$

$$\geq n_i - \sum_{j=1}^{n_i} \sum_{\ell=1}^{b_{ij}} p_{i,j,\ell} \frac{\tau}{v_{i,j,\ell}} = r^-(t, D_i). \qquad (6)$$

Markov Inequality Lower Bound

- We know that the pdf of any unseen t must satisfy $E[X] \leq \tau$
- We can use the **Markov Inequality** to lower bound the rank of any site s_i with database D_i as,

$$r(t, D_i) = \sum_{j=1}^{n_i} \Pr[X_j > X] = n_i - \sum_{j=1}^{n_i} \Pr[X \geq X_j]$$

$$\geq n_i - \sum_{j=1}^{n_i} \sum_{\ell=1}^{b_{ij}} p_{i,j,\ell} \frac{E[X]}{v_{i,j,\ell}}. \qquad \text{(Markov Ineq.)}$$

$$\geq n_i - \sum_{j=1}^{n_i} \sum_{\ell=1}^{b_{ij}} p_{i,j,\ell} \frac{\tau}{v_{i,j,\ell}} = r^-(t, D_i). \qquad (6)$$

- Now the global rank $r(t)$ must satisfy

$$r(t) \geq \sum_{i=1}^{m} r^-(t, D_i) = r_\lambda^- \qquad (7)$$

Markov Inequality Lower Bound

- We know that the pdf of any unseen t must satisfy $E[X] \leq \tau$
- We can use the **Markov Inequality** to lower bound the rank of any site s_i with database D_i as,

$$r(t, D_i) = \sum_{j=1}^{n_i} \Pr[X_j > X] = n_i - \sum_{j=1}^{n_i} \Pr[X \geq X_j]$$

Loose!

$$\geq n_i - \sum_{j=1}^{n_i} \sum_{\ell=1}^{b_{ij}} p_{i,j,\ell} \frac{E[X]}{v_{i,j,\ell}}. \qquad \text{(Markov Ineq.)}$$

$$\geq n_i - \sum_{j=1}^{n_i} \sum_{\ell=1}^{b_{ij}} p_{i,j,\ell} \frac{\tau}{v_{i,j,\ell}} = r^-(t, D_i). \qquad (6)$$

- Now the global rank $r(t)$ must satisfy

$$r(t) \geq \sum_{i=1}^{m} r^-(t, D_i) = r_\lambda^- \qquad (7)$$

Linear Programming Lower Bound

- Any unseen tuple t must have $E[X] \leq \tau$

Linear Programming Lower Bound

- Any unseen tuple t must have $E[X] \leq \tau$
- We've seen how to derive a lower bound r_λ^- on the global rank for any unseen tuple t using Markov's Inequality

Linear Programming Lower Bound

- Any unseen tuple t must have $E[X] \leq \tau$
- We've seen how to derive a lower bound r_λ^- on the global rank for any unseen tuple t using Markov's Inequality
- We want to find as tight a r_λ^- as possible by finding the smallest possible $r^-(t, D_i)$'s at each site

Linear Programming Lower Bound

- Any unseen tuple t must have $E[X] \leq \tau$
- We've seen how to derive a lower bound r_λ^- on the global rank for any unseen tuple t using Markov's Inequality
- We want to find as tight a r_λ^- as possible by finding the smallest possible $r^-(t, D_i)$'s at each site
- We can use Linear Programming in order to derive the $r^-(t, D_i)$ at each site to find a tight r_λ^-

Linear Programming

- The idea is to construct the best possible X for an unseen tuple t at each site s_i that obtains the smallest possible local rank for each s_i

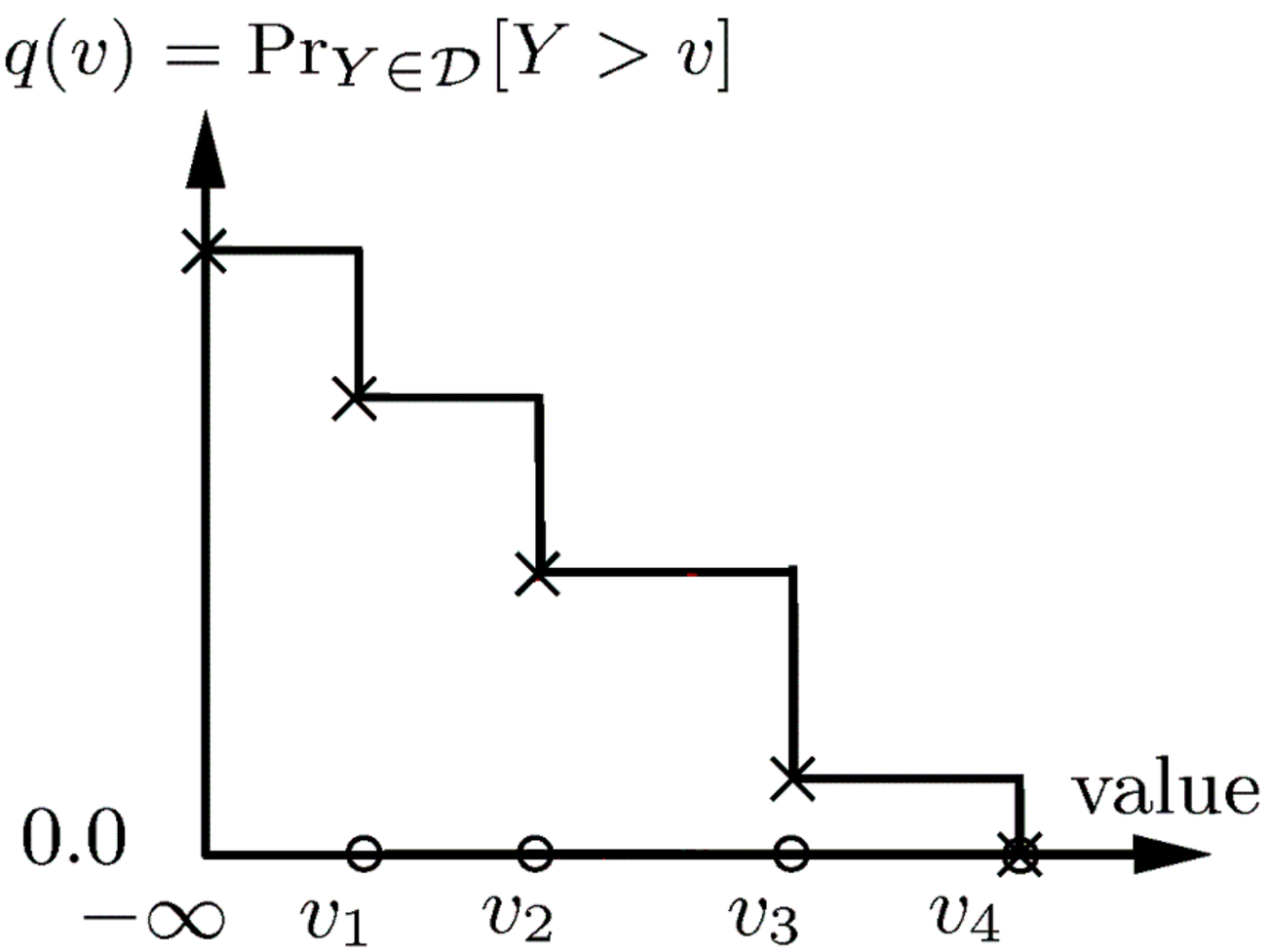
Linear Programming

- The idea is to construct the best possible X for an unseen tuple t at each site s_i that obtains the smallest possible local rank for each s_i
- X could take on arbitrary v_ℓ's as it's possible score values, some of which do not exist in value universe U_i at a site s_i

Linear Programming

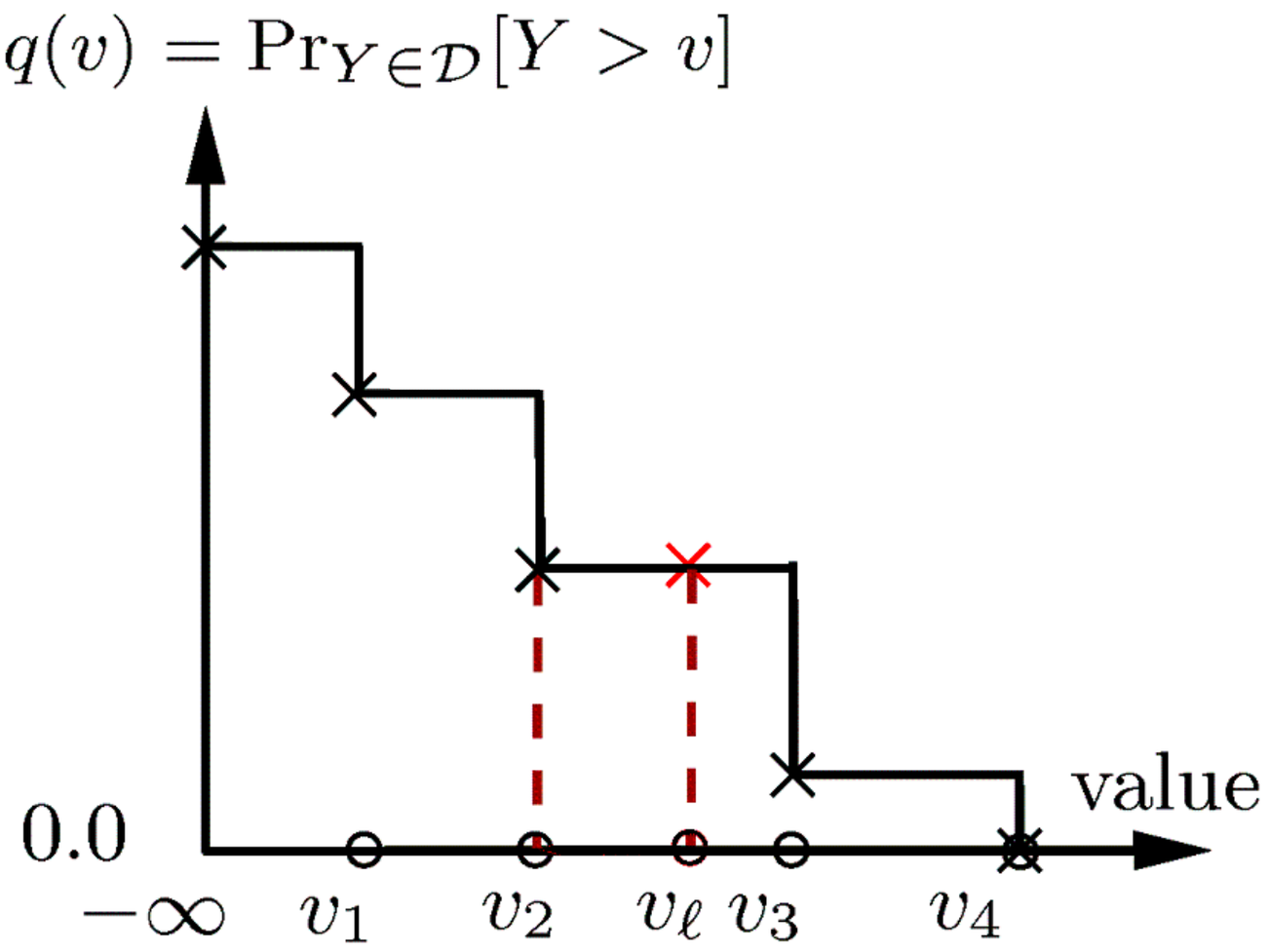
- The idea is to construct the best possible X for an unseen tuple t at each site s_i that obtains the smallest possible local rank for each s_i
- X could take on arbitrary v_ℓ's as it's possible score values, some of which do not exist in value universe U_i at a site s_i
- We can show this problem is irrelevant after studying the semantics of the $r(t, D_i)$'s and the $q(v)$'s

Linear Programming: a Note on $q(v)$'s



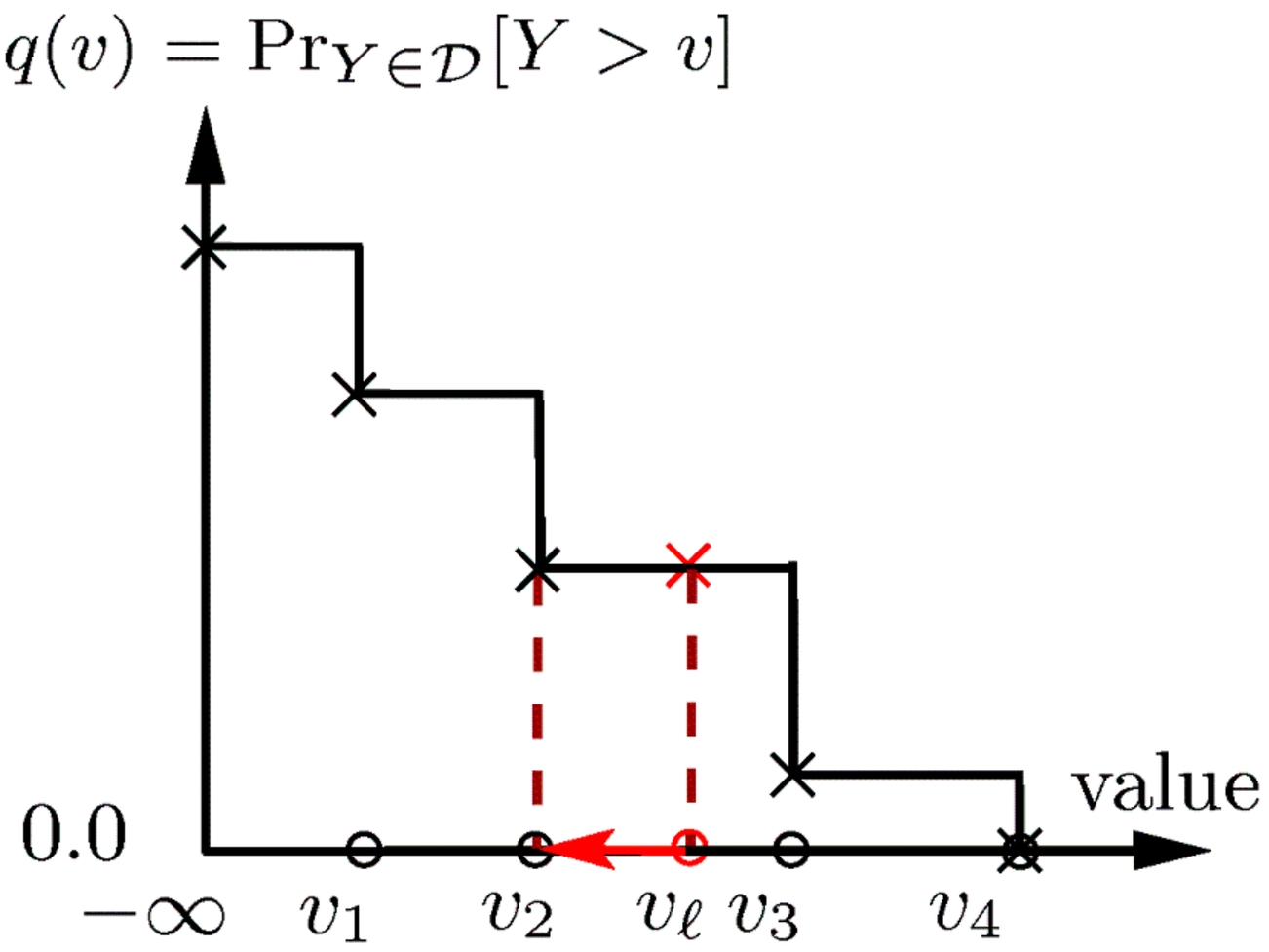
- Recall that $r(t_{i,j}, D_y) = \sum_{\ell=1}^{b_{i,j}} p_{i,j,l} q_y(v_{i,j,l})$ and $q(v)$ is essentially a stair case curve as above

Linear Programming



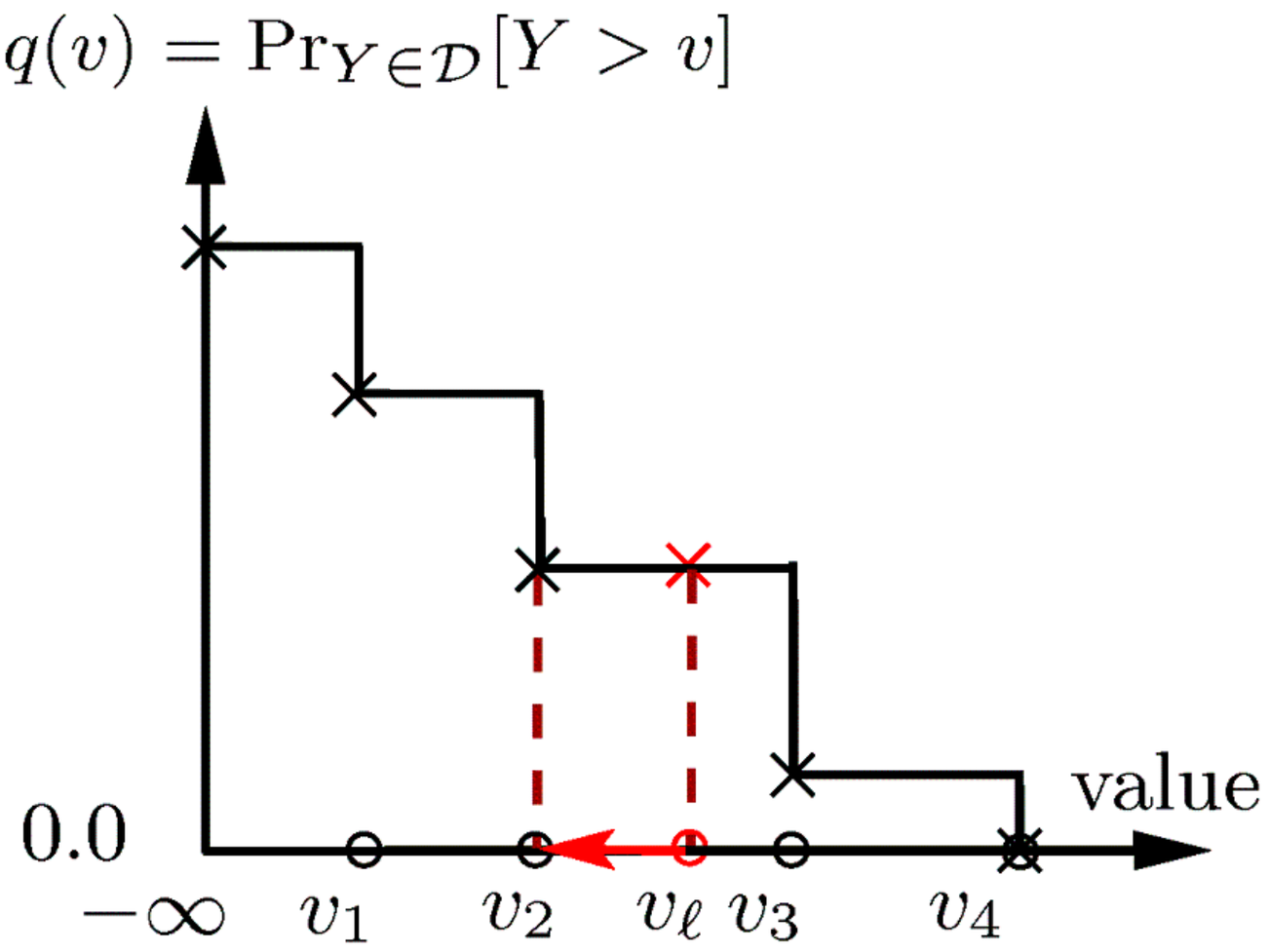
- X may take a value v_ℓ not in U_i with v_2 as its nearest left neighbor

Linear Programming



- Even if X takes a value v_ℓ not in U_i we can decrease v_ℓ until we hit v_2 in U_i and $E[X] \leq \tau$ clearly still holds as we are only decreasing the value of one of the choices in X

Linear Programming



- Also note that during this transformation $q(v_\ell) = q(v_2)$ and so the **local rank** of t remains the same

Linear Programming Formulation

- Now we can assume X draws values from U_i

Linear Programming Formulation

- Now we can assume X draws values from U_i
- Then we can define a linear program with the constraints

$$0 \leq p_\ell \leq 1$$
$$\ell = 1, \ldots, \gamma = |U_i|$$
$$p_1 + \ldots + p_\gamma = 1$$
$$p_1 v_1 + \ldots + p_\gamma v_\gamma \leq \tau$$

and minimize the local rank which is,

$$r(X, D_i) = \sum_{\ell=1}^{\gamma} p_\ell q_i(v_\ell)$$

Linear Programming Formulation

- Now we can assume X draws values from U_i
- Then we can define a linear program with the constraints

$$0 \leq p_\ell \leq 1$$
$$\ell = 1, \ldots, \gamma = |U_i|$$
$$p_1 + \ldots + p_\gamma = 1$$
$$p_1 v_1 + \ldots + p_\gamma v_\gamma \leq \tau$$

and minimize the local rank which is,

$$r(X, D_i) = \sum_{\ell=1}^{\gamma} p_\ell q_i(v_\ell)$$

- Each site can conduct these linear programs at each round in order for the server to derive a tight r_λ^-

Linear Programming Formulation

- Now we can assume X draws values from U_i
- Then we can define a linear program with the constraints

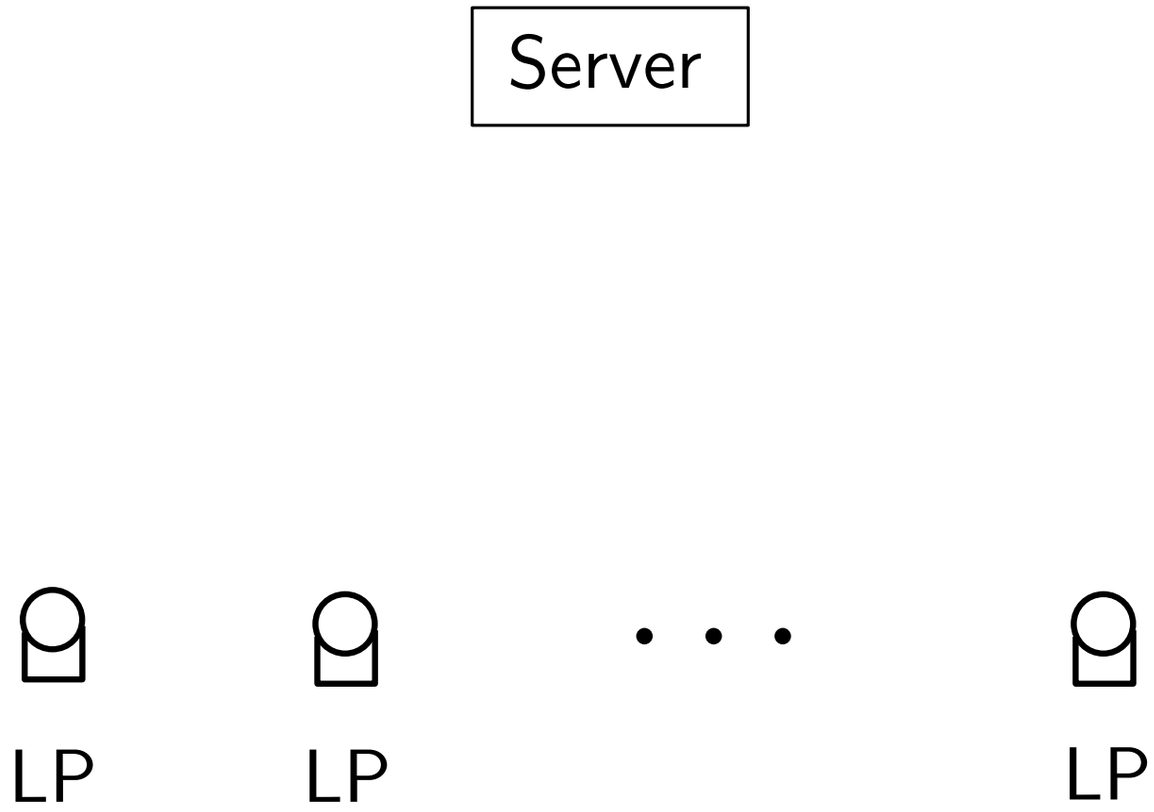
$$0 \leq p_\ell \leq 1$$
$$\ell = 1, \ldots, \gamma = |U_i|$$
$$p_1 + \ldots + p_\gamma = 1$$
$$p_1 v_1 + \ldots + p_\gamma v_\gamma \leq \tau$$

and minimize the local rank which is,

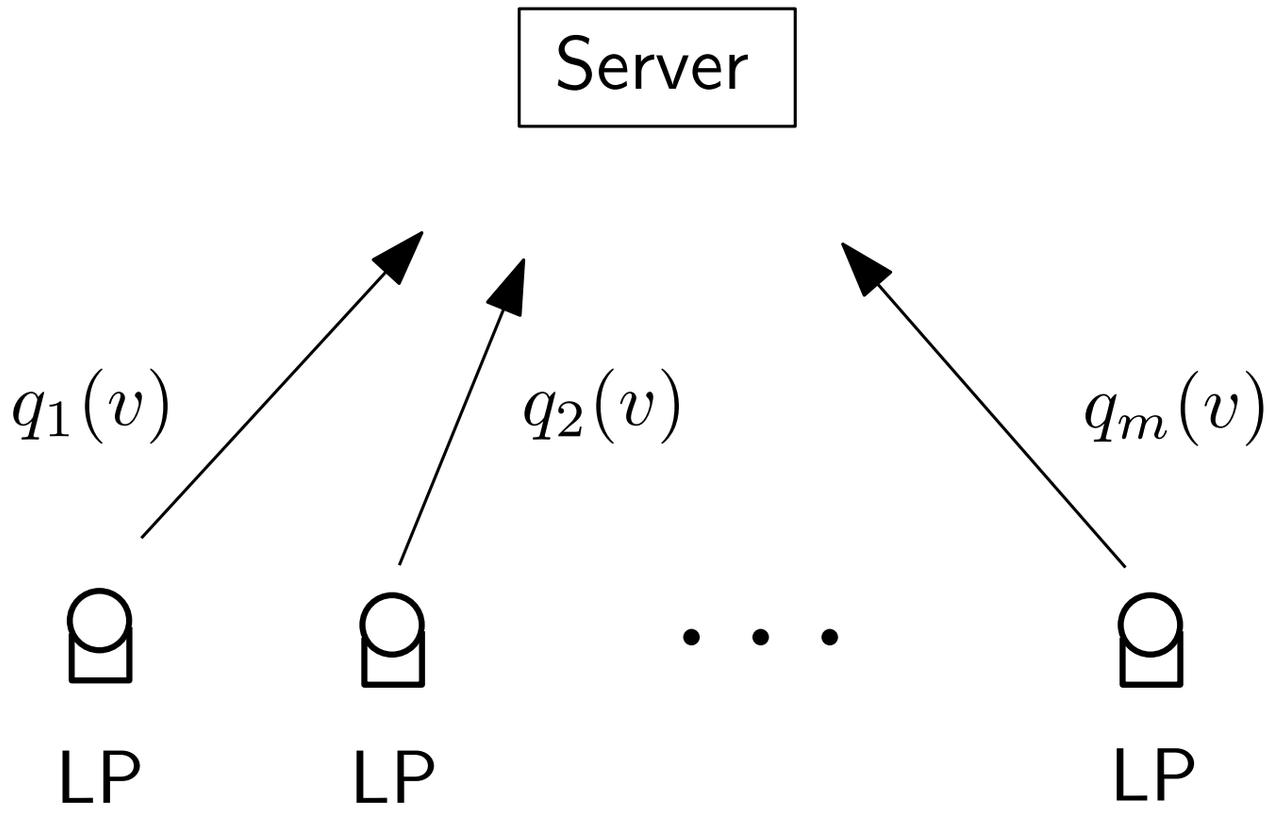
$$r(X, D_i) = \sum_{\ell=1}^{\gamma} p_\ell q_i(v_\ell)$$

- Each site can conduct these linear programs at each round in order for the server to derive a tight r_λ^-
- This algorithm is denoted as $A - LP$

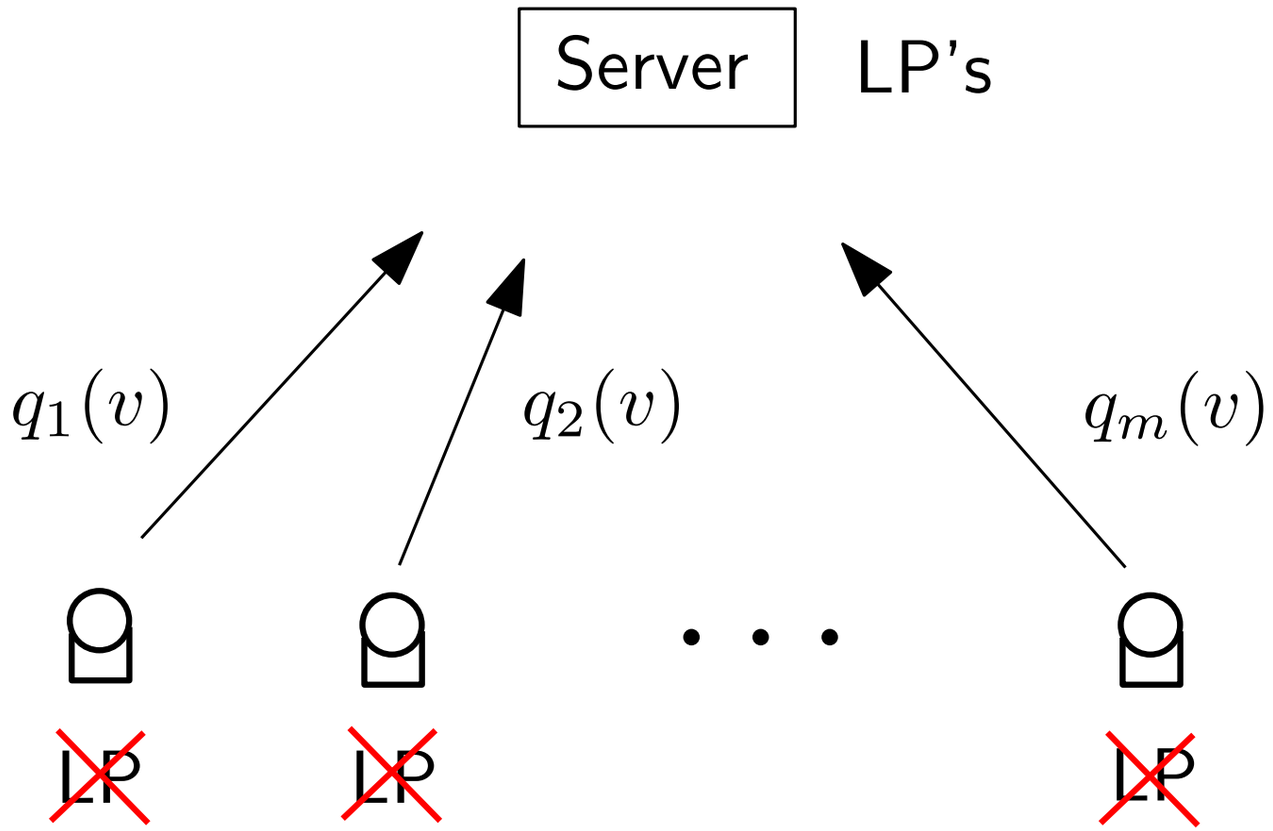
Eliminating LP's at distributed sites



Eliminating LP's at distributed sites

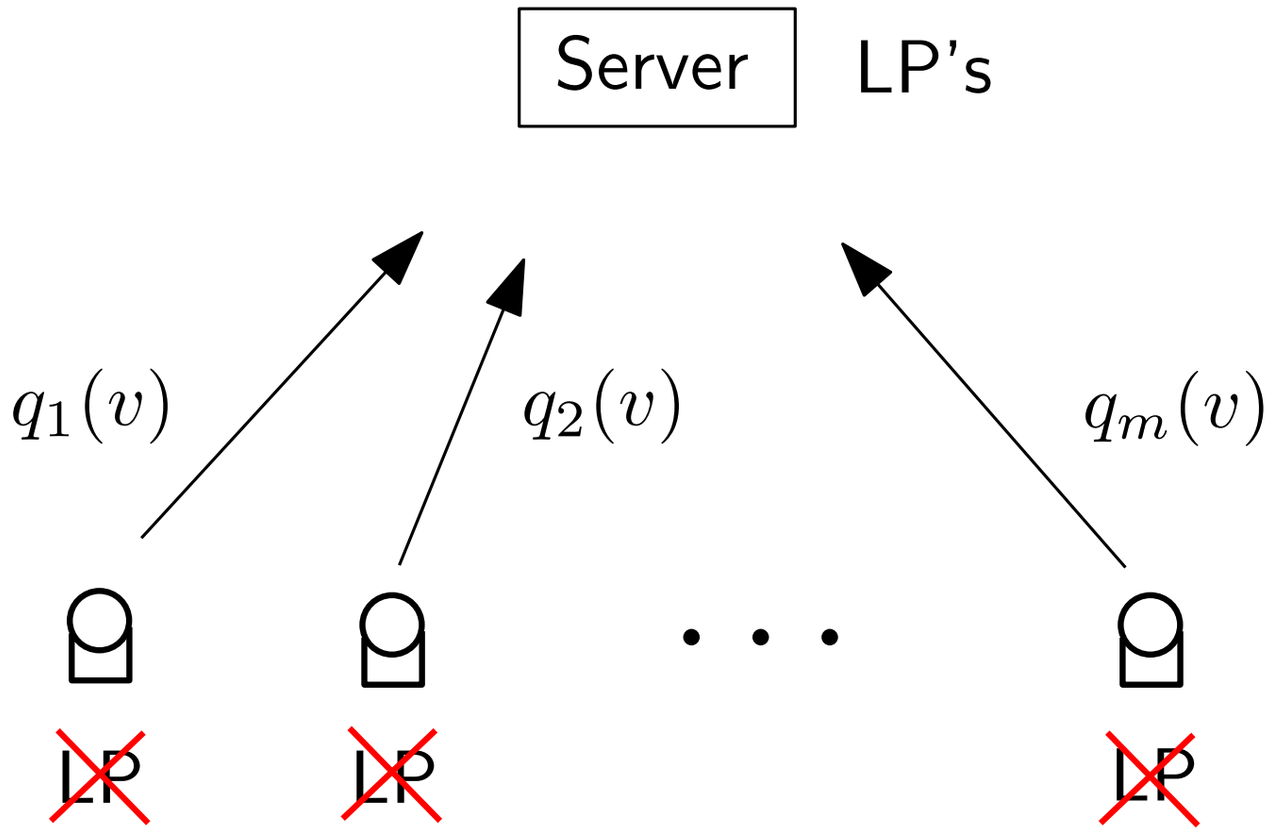


Eliminating LP's at distributed sites

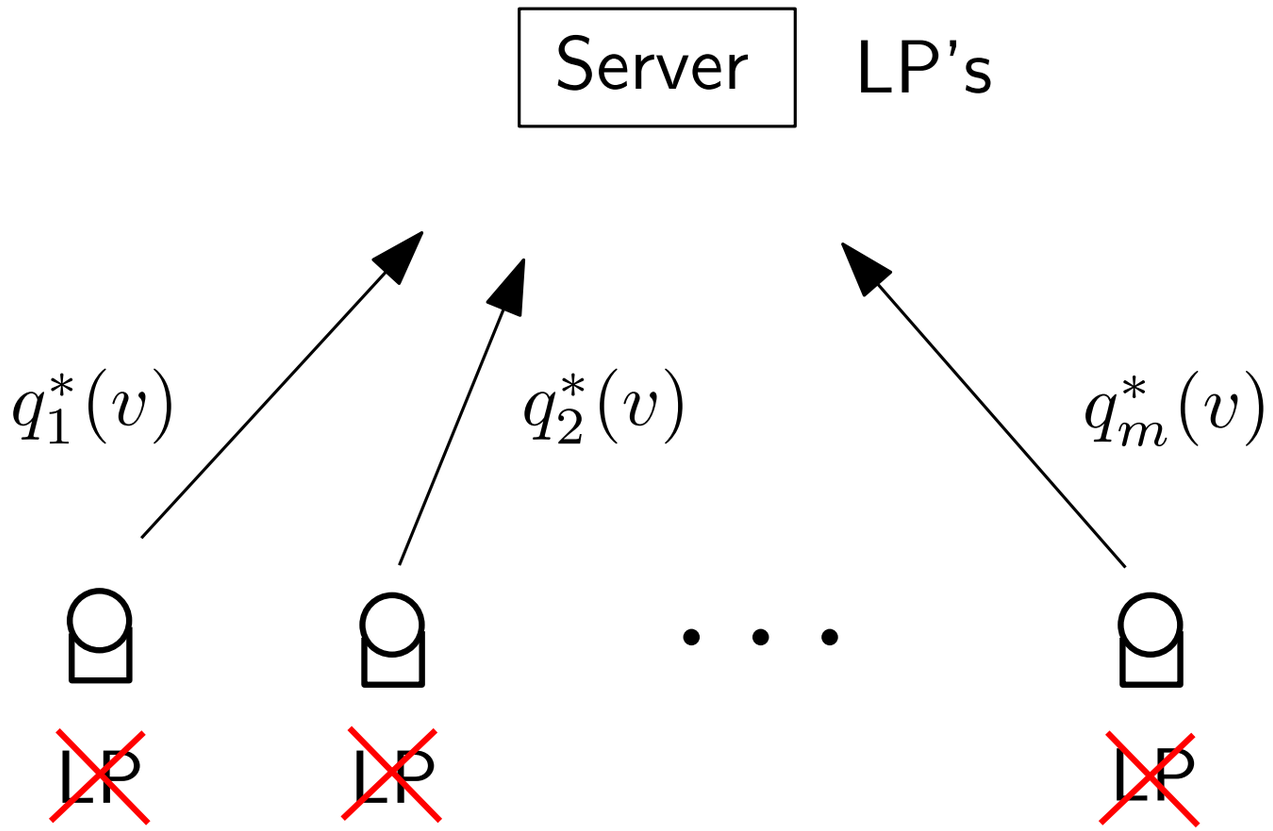


Eliminating LP's at distributed sites

Comunication expensive!



Eliminating LP's at distributed sites



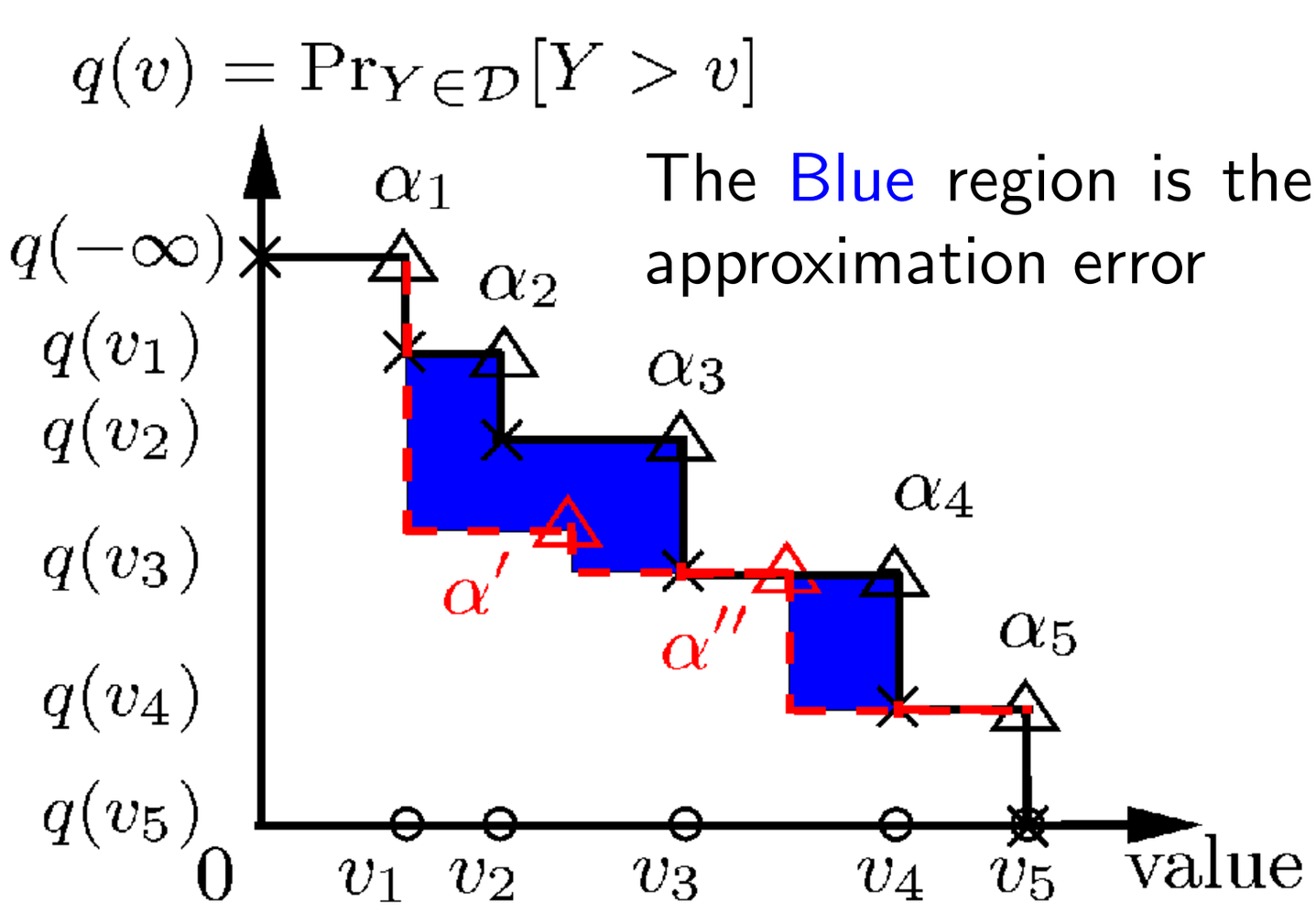
$q^*(v)$'s : the approximate $q(v)$'s

- Formally, the problem is to find the optimal approximation $q^*(v)$ to a $q(v)$ which obtains the smallest approximation error given a fixed budget η

$q^*(v)$'s : the approximate $q(v)$'s

- Formally, the problem is to find the optimal approximation $q^*(v)$ to a $q(v)$ which obtains the smallest approximation error given a fixed budget η
- We also must ensure that by using these $q^*(v)$'s we still arrive at the actual **global top-k** at the server

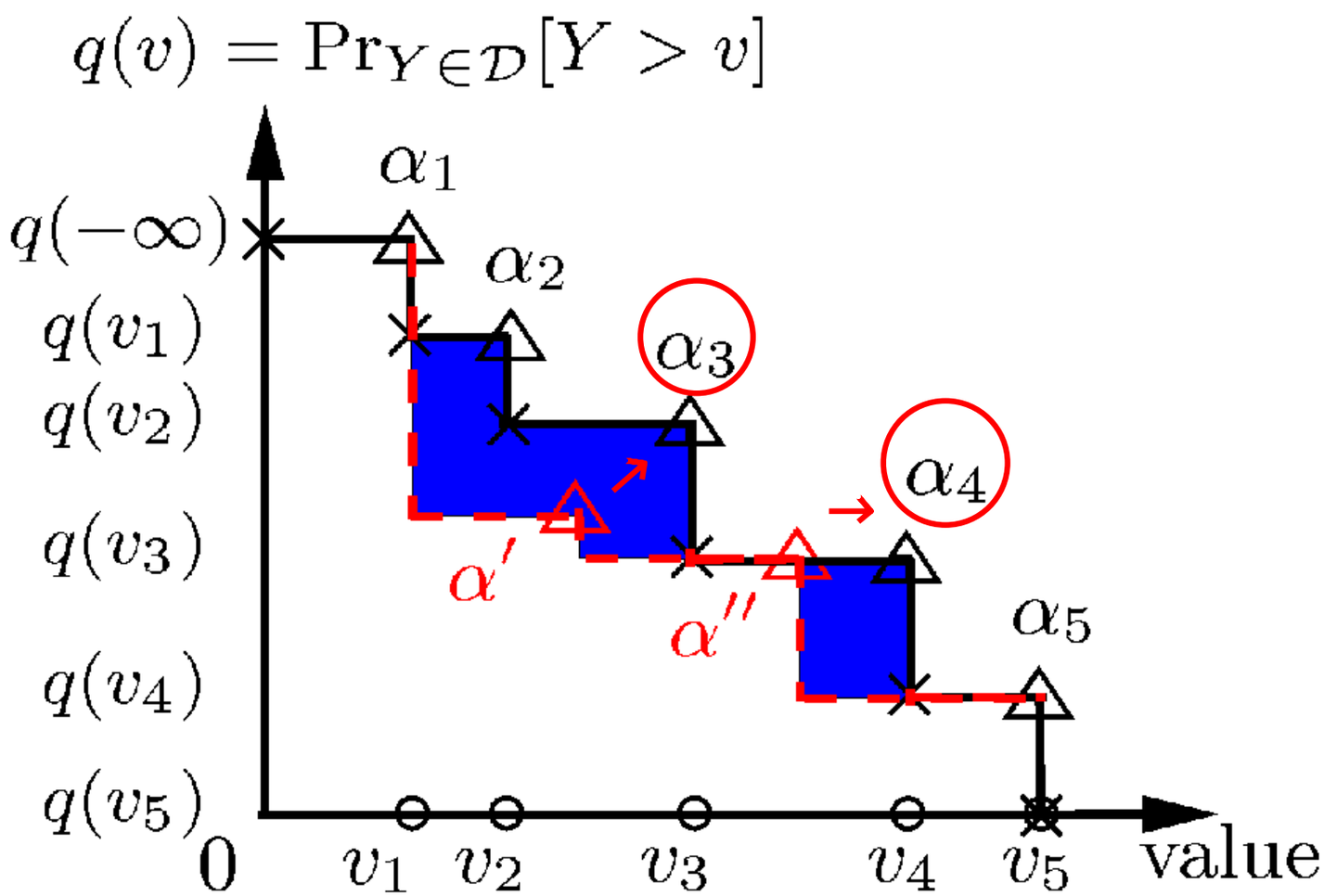
$q^*(v)$'s the approximate $q(v)$'s



The Blue region is the approximation error

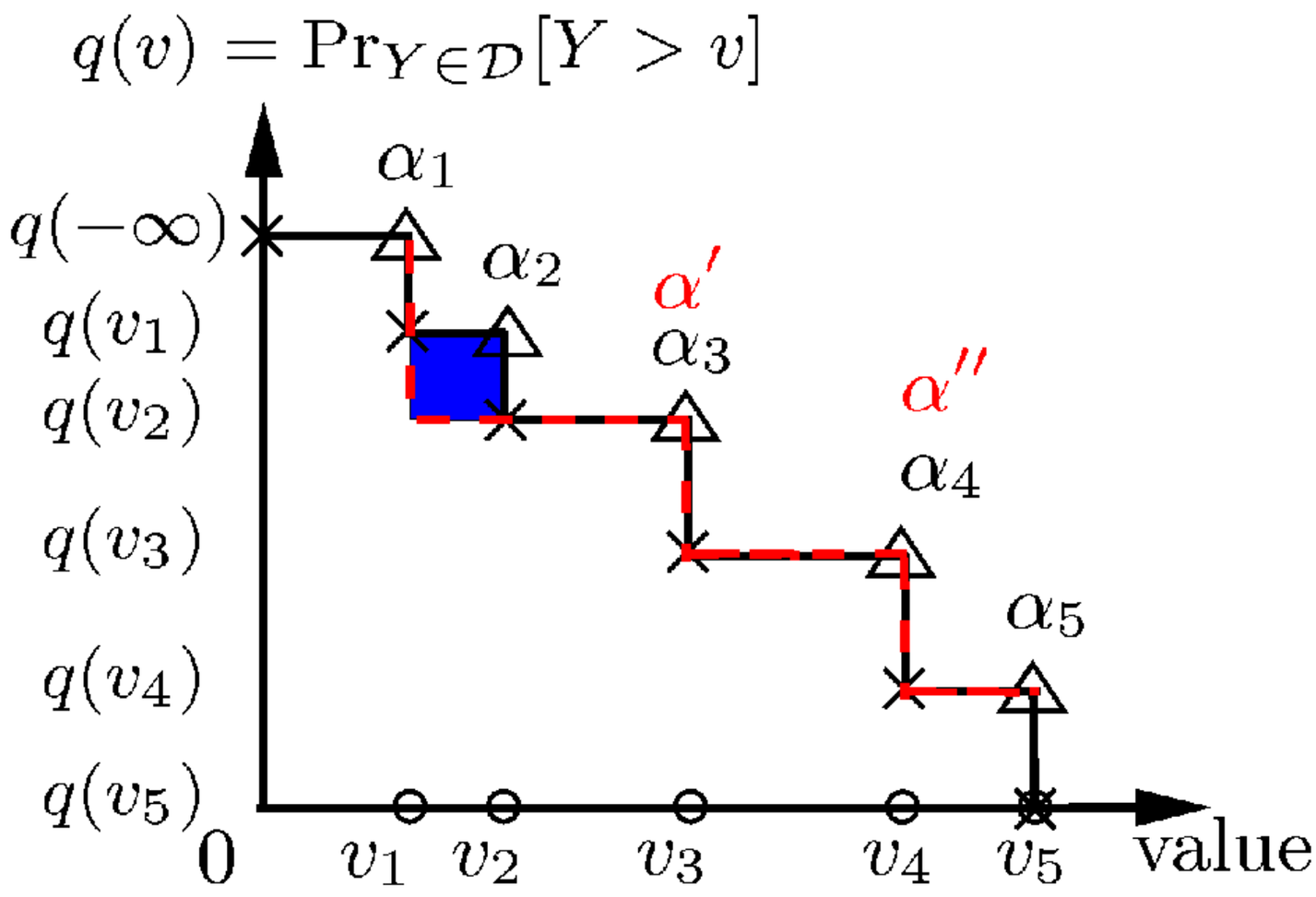
- Above we see a $q^*(v)$ which takes two points α' and α'' which are not right upper corner points in the original $q(v)$

$q^*(v)$'s the approximate $q(v)$'s



- We can minimize the error between the $q^*(v)$ curve and $q(v)$ curve by sampling only the right upper corner points

$q^*(v)$'s the approximate $q(v)$'s



- The new error after selecting α' as α_3 and α'' as α_4 is shown by the blue region

$q^*(v)$'s the approximate $q(v)$'s

- In order to find the optimal $q^*(v)$ for a $q(v)$ we can formulate a dynamic program

$$A(i,j) = min \begin{cases} min_{x \in [i-1,j-1]} \{A(i-1,x) - \delta^j_{q^*(i-1,x)}\} \\ min_{x \in [i,j-1]} \{A(i,x)\} \end{cases} \quad (8)$$

$q^*(v)$'s the approximate $q(v)$'s

- In order to find the optimal $q^*(v)$ for a $q(v)$ we can formulate a dynamic program

$$A(i,j) = min \begin{cases} min_{x \in [i-1, j-1]} \{A(i-1,x) - \delta^j_{q^*(i-1,x)}\} \\ min_{x \in [i, j-1]} \{A(i,x)\} \end{cases} \quad (8)$$

- $A(i,j)$ is the optimal approximation error from selecting i points from the first j points from $q(v)$

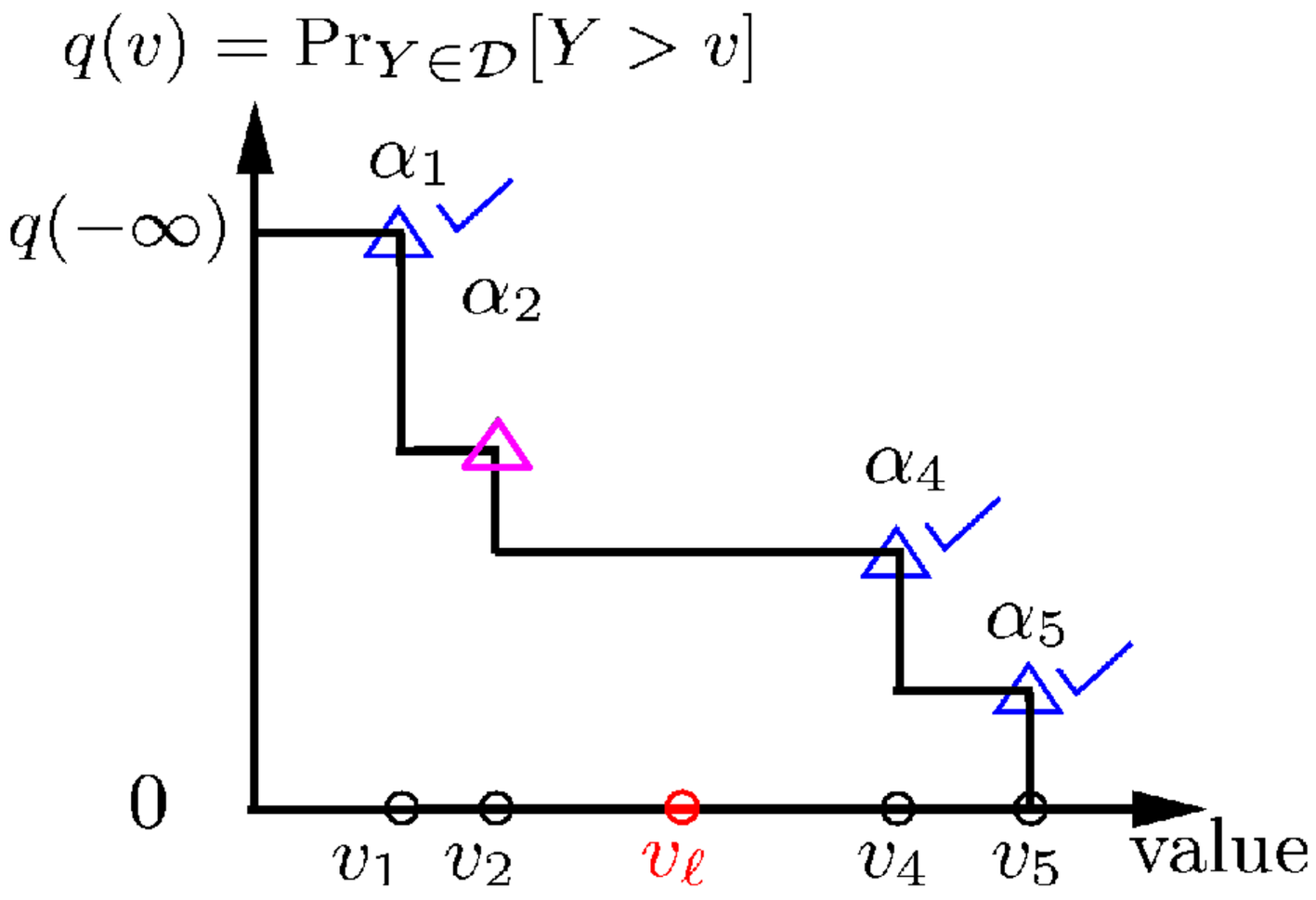
$q^*(v)$'s the approximate $q(v)$'s

- In order to find the optimal $q^*(v)$ for a $q(v)$ we can formulate a dynamic program

$$A(i,j) = min \begin{cases} min_{x \in [i-1,j-1]} \{A(i-1,x) - \delta^j_{q^*(i-1,x)}\} \\ min_{x \in [i,j-1]} \{A(i,x)\} \end{cases} \quad (8)$$

- $A(i,j)$ is the optimal approximation error from selecting i points from the first j points from $q(v)$
- This algorithm is denoted as $A - ALP$

Updating the $q^*(v)$'s at the server.... for free



$$+ \ (v_\ell, p_\ell)$$

- During some round when the server retrieves a tuple t with pdf X from a site s_i to update the representative queue, the server may see a (v_ℓ, p_ℓ) pair in X s.t. v_ℓ was not an originally sampled upper right corner point from $q(v)$

Other Issues.... Latency

- Currently we check the termination condition at the end of every round

Other Issues.... Latency

- Currently we check the termination condition at the end of every round
- An intuitive idea is that we may reduce latency by checking termination condition only after every β rounds

Other Issues.... Latency

- Currently we check the termination condition at the end of every round
- An intuitive idea is that we may reduce latency by checking termination condition only after every β rounds
- This will reduce the computational burden at the server for the $A - ALP$ algorithm, and reduce the computational burden at the sites for the $A - LP$ algorithms

Other Issues.... Latency

- Currently we check the termination condition at the end of every round
- An intuitive idea is that we may reduce latency by checking termination condition only after every β rounds
- This will reduce the computational burden at the server for the $A-ALP$ algorithm, and reduce the computational burden at the sites for the $A-LP$ algorithms
- The tradeoff is that we could potentially miss the optimal termination point, but not by more than β tuples

Experimental Setup

- Conducted on an Intel Xeon 5130 CPU @ 2GHz with 4GB memory

Experimental Setup

- Conducted on an Intel Xeon 5130 CPU @ 2GHz with 4GB memory
- We utilized three real data sets
  - Movie data set from the Mystiq project containing 56,000 records
  - Temperature data set collected from 54 sensors from the Intel Research Berkley lab containing 64,000 records
  - Chlorine data set from the EPANET project containing 67,000 records

Experimental Setup

- Conducted on an Intel Xeon 5130 CPU @ 2GHz with 4GB memory
- We utilized three real data sets
  - Movie data set from the Mystiq project containing 56,000 records
  - Temperature data set collected from 54 sensors from the Intel Research Berkley lab containing 64,000 records
  - Chlorine data set from the EPANET project containing 67,000 records
- We utilized one synthetic data set
  - Synthetic Gaussian where each record's score attribute draws it's values from a Gaussian distribution with standard deviation [1, 1000] and the mean [5 * σ, 100000]

Experimental Setup

- The default experimental parameters are summarized below
Symbol	Definition	Default Value
N	number of tuples	56,000
$\|X\|$	choices in a tuple's pdf	5
m	number of sites	10
k	number of tuples to rank	100
η	$\|q^*(v)\|$	$1\% \times \|q(v)\|$

Experimental Setup

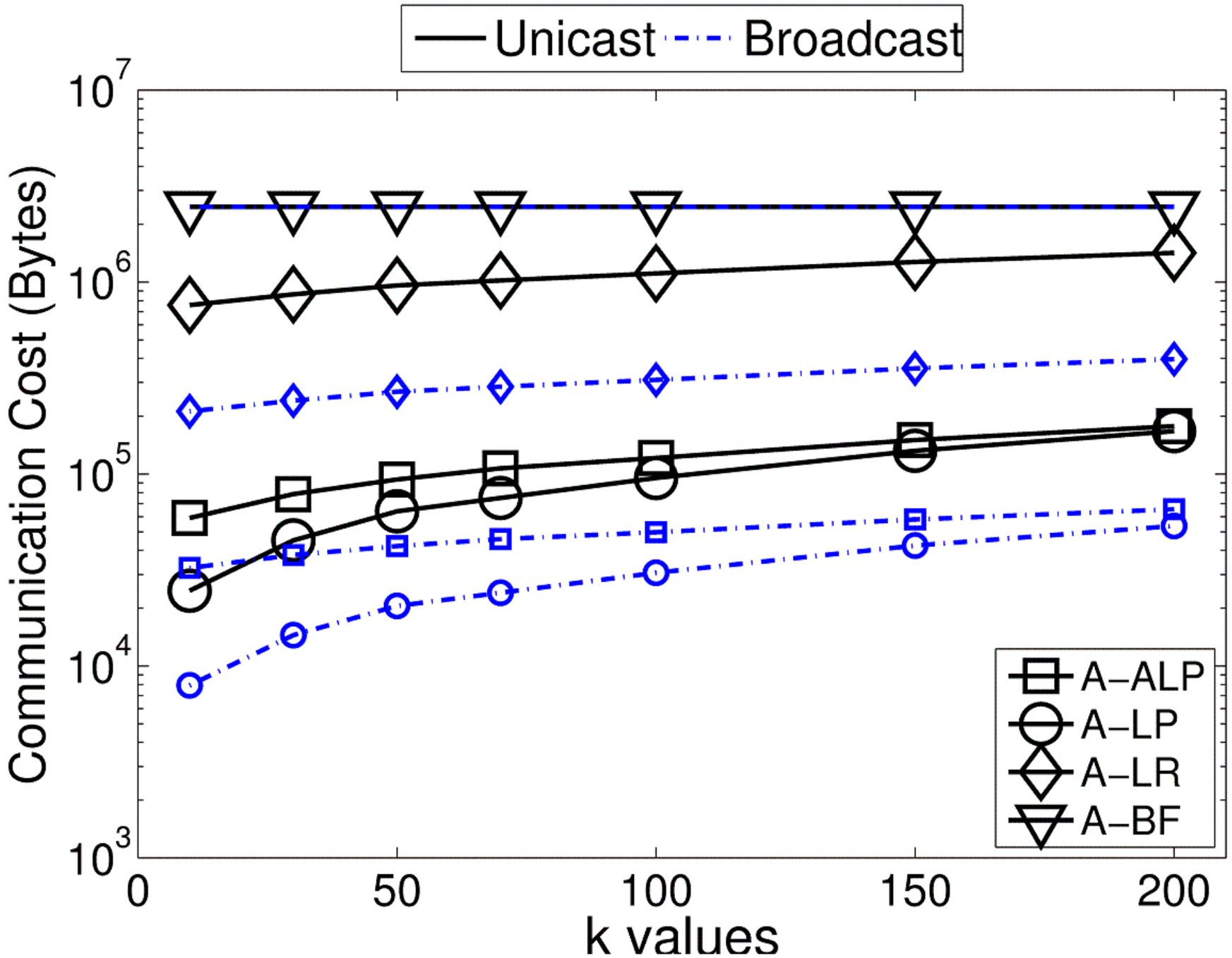
- The default experimental parameters are summarized below

Symbol	Definition	Default Value
N	number of tuples	56,000
$\|X\|$	choices in a tuple's pdf	5
m	number of sites	10
k	number of tuples to rank	100
η	$\|q^*(v)\|$	$1\% \times \|q(v)\|$

- In addition communication costs are determined as follows

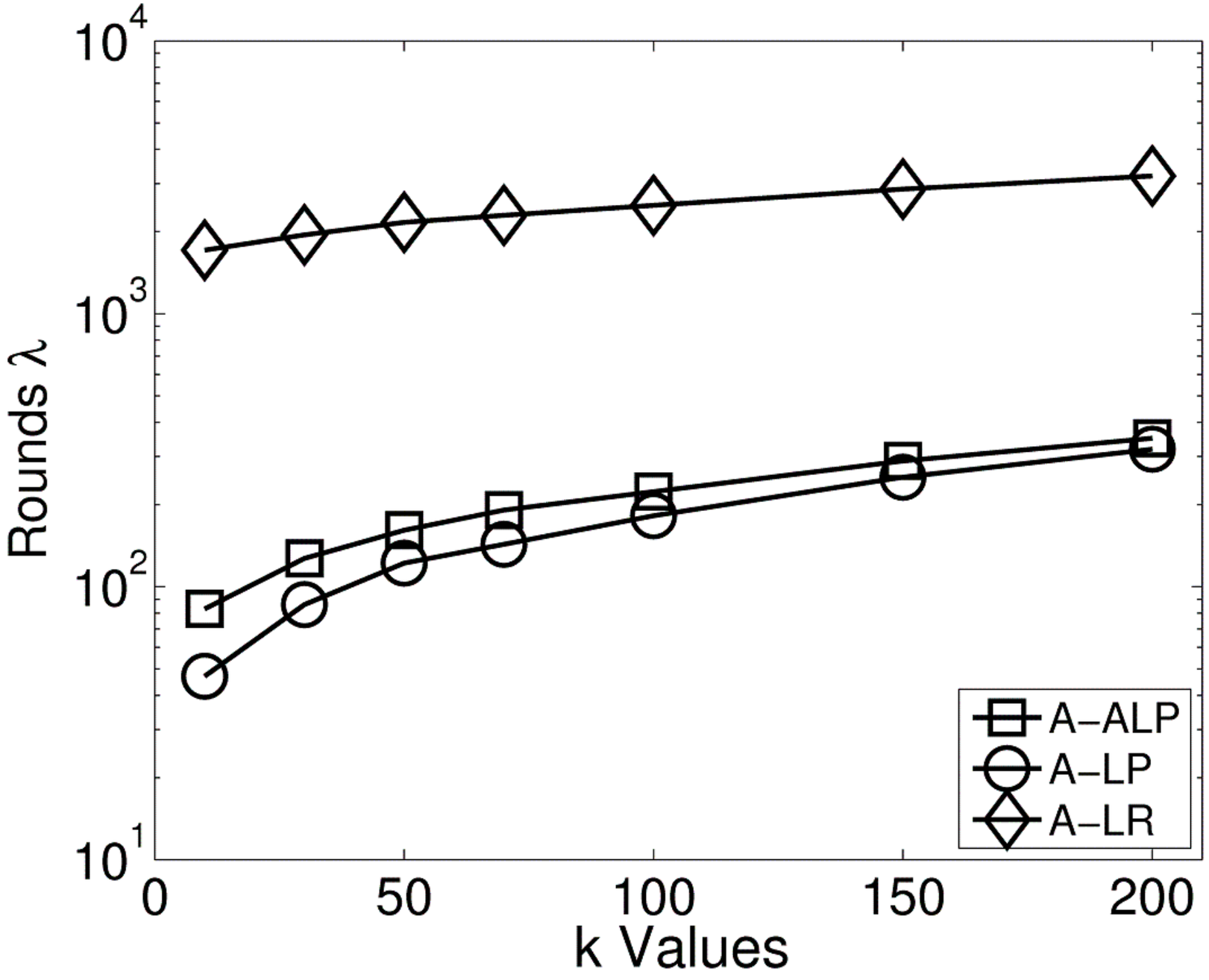
Object	Definition	Communication Cost (bytes)
v	value	4
p	probability	4
X	pdf	$\|X\| \times 8$
t	tuple	$(\|X\| \times 8) + 4$
$r(t, D_i)$	local rank of t in D_i	4
$E[X]$	Expectance of X	4

Communication Cost as k Varies



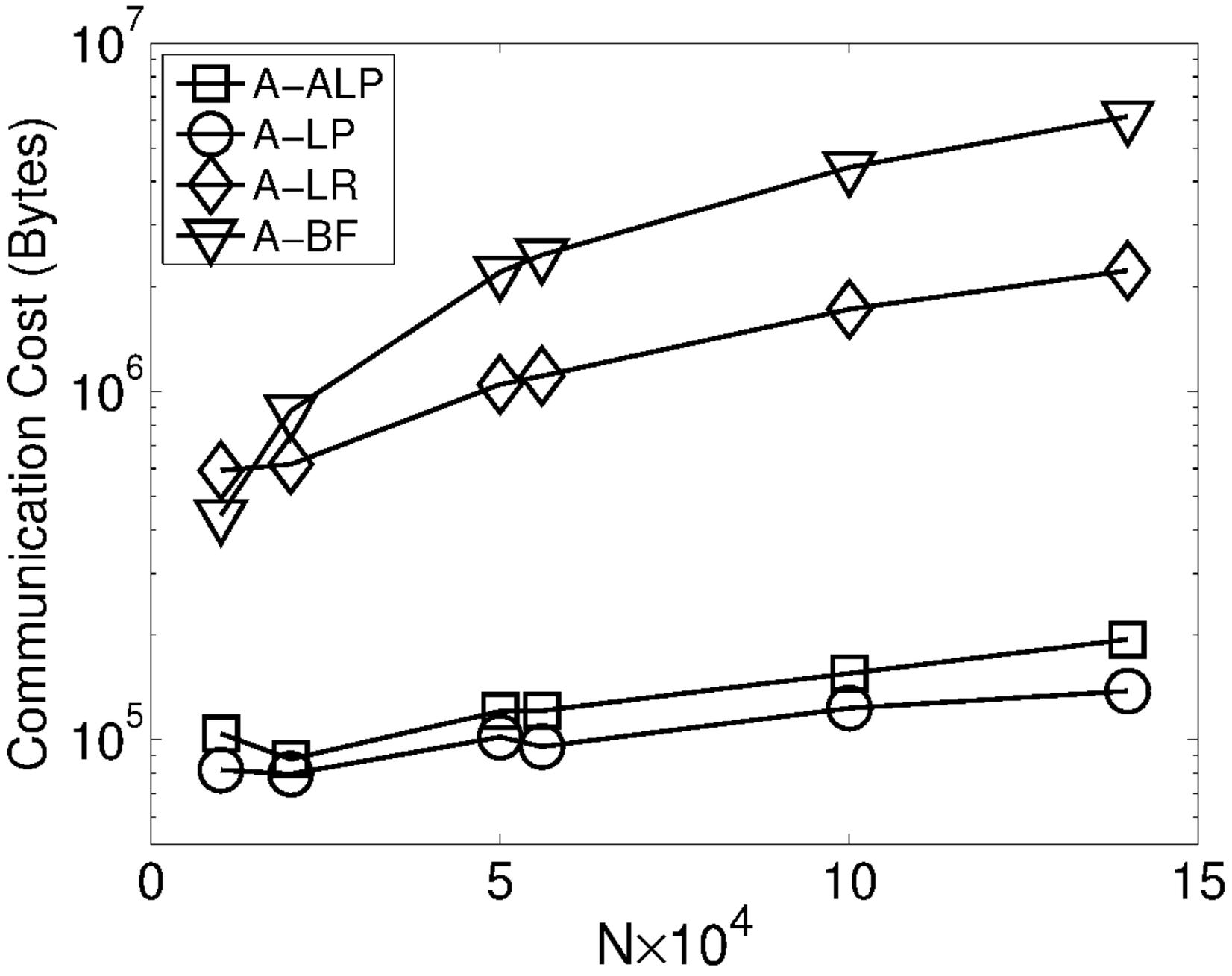
Chlorine Data Set

Number of Rounds as k Varies



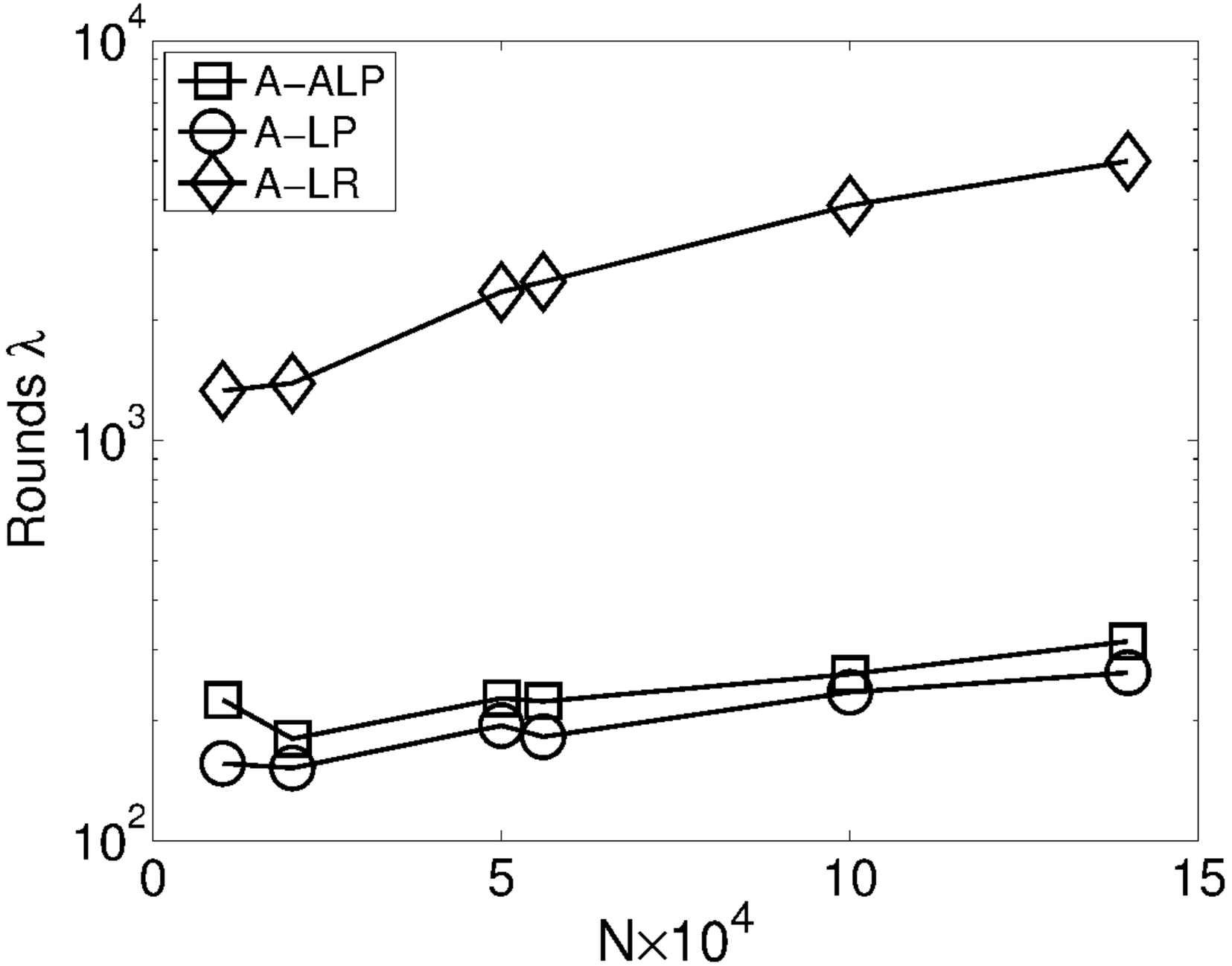
Chlorine Data Set

Effect of N on Communication Cost



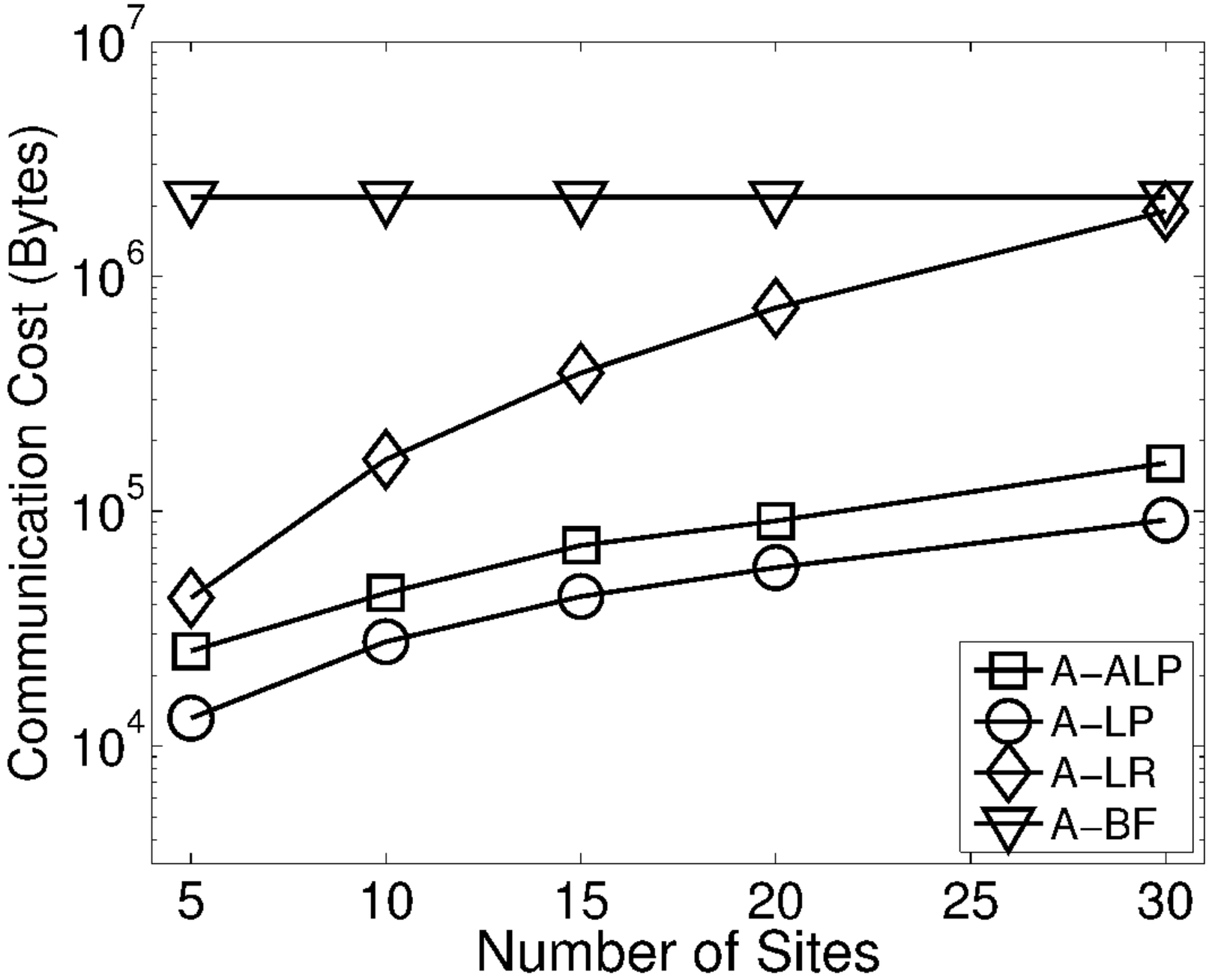
Chlorine Data Set

Effect of N on Number of Rounds



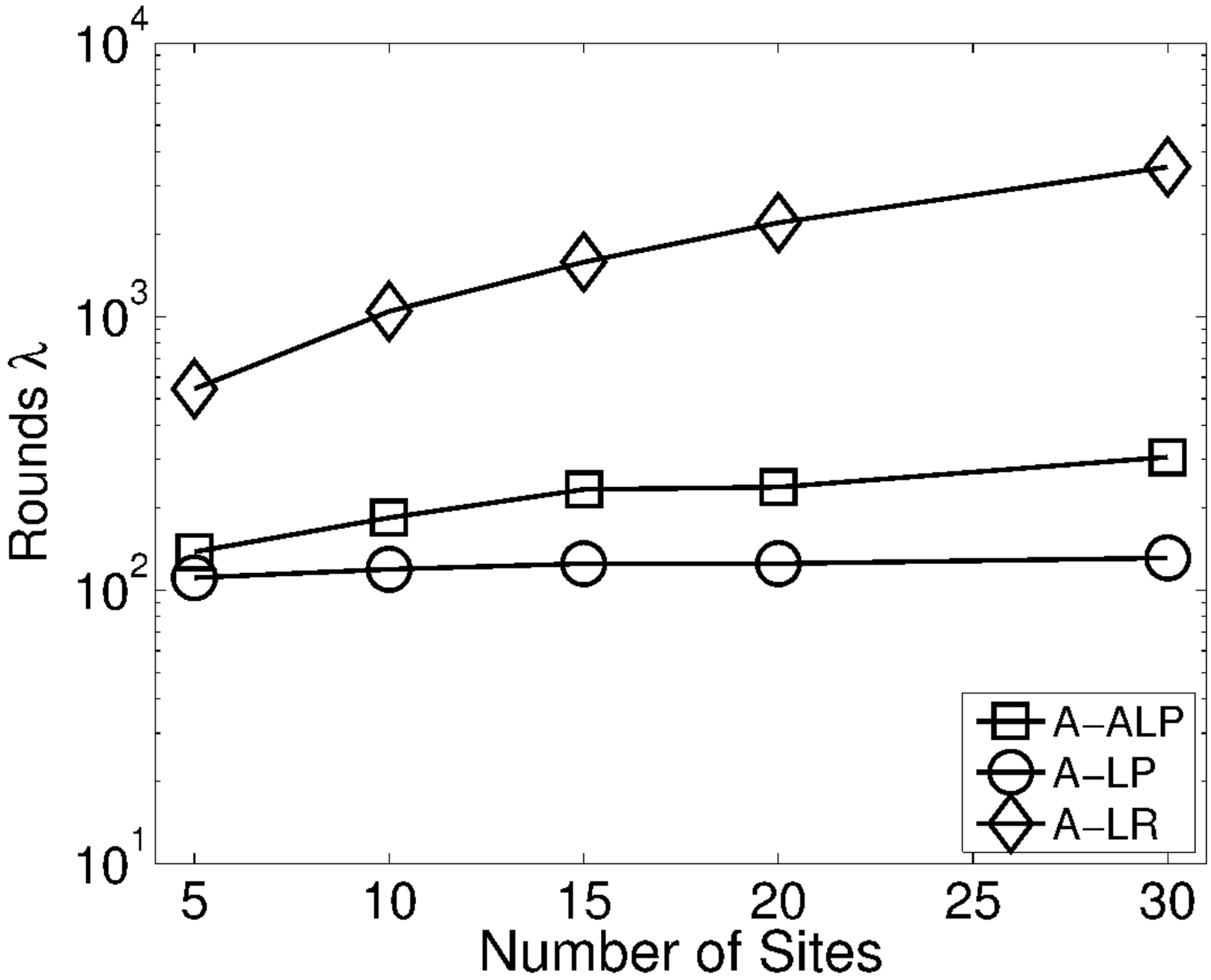
Chlorine Data Set

Effect of m on Communication Cost



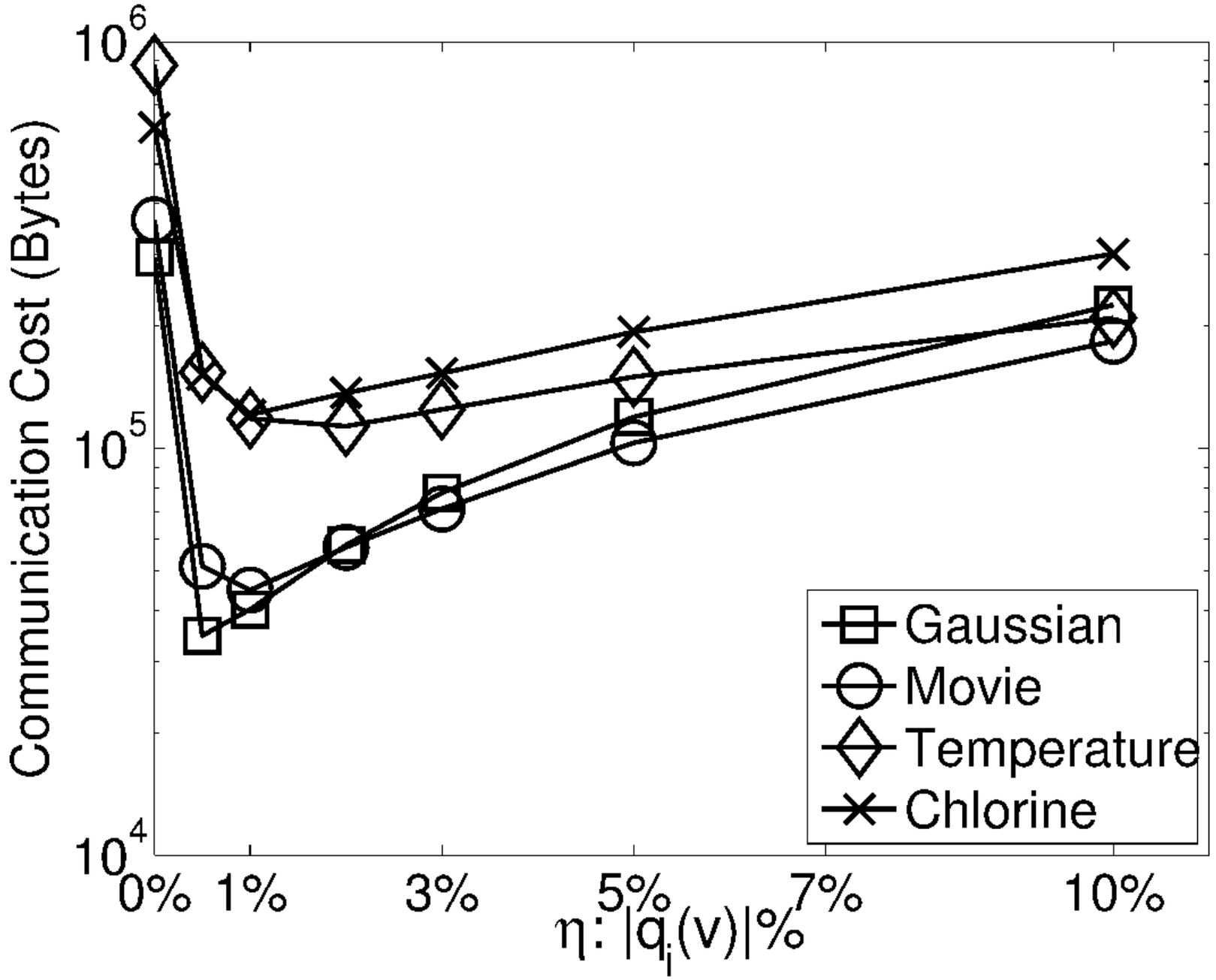
Chlorine Data Set

Effect of m on Number of Rounds



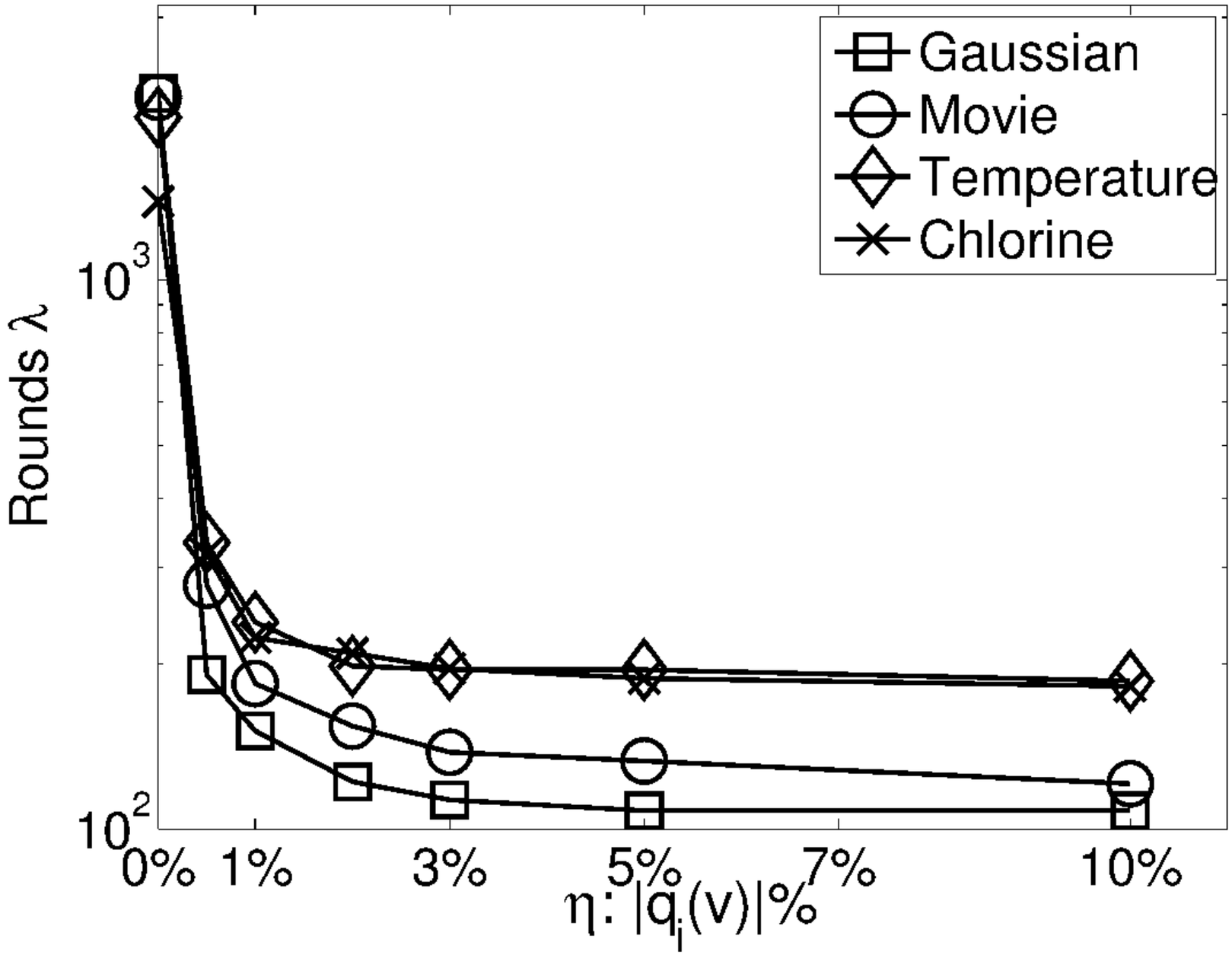
Chlorine Data Set

Effect of η on Communication Cost



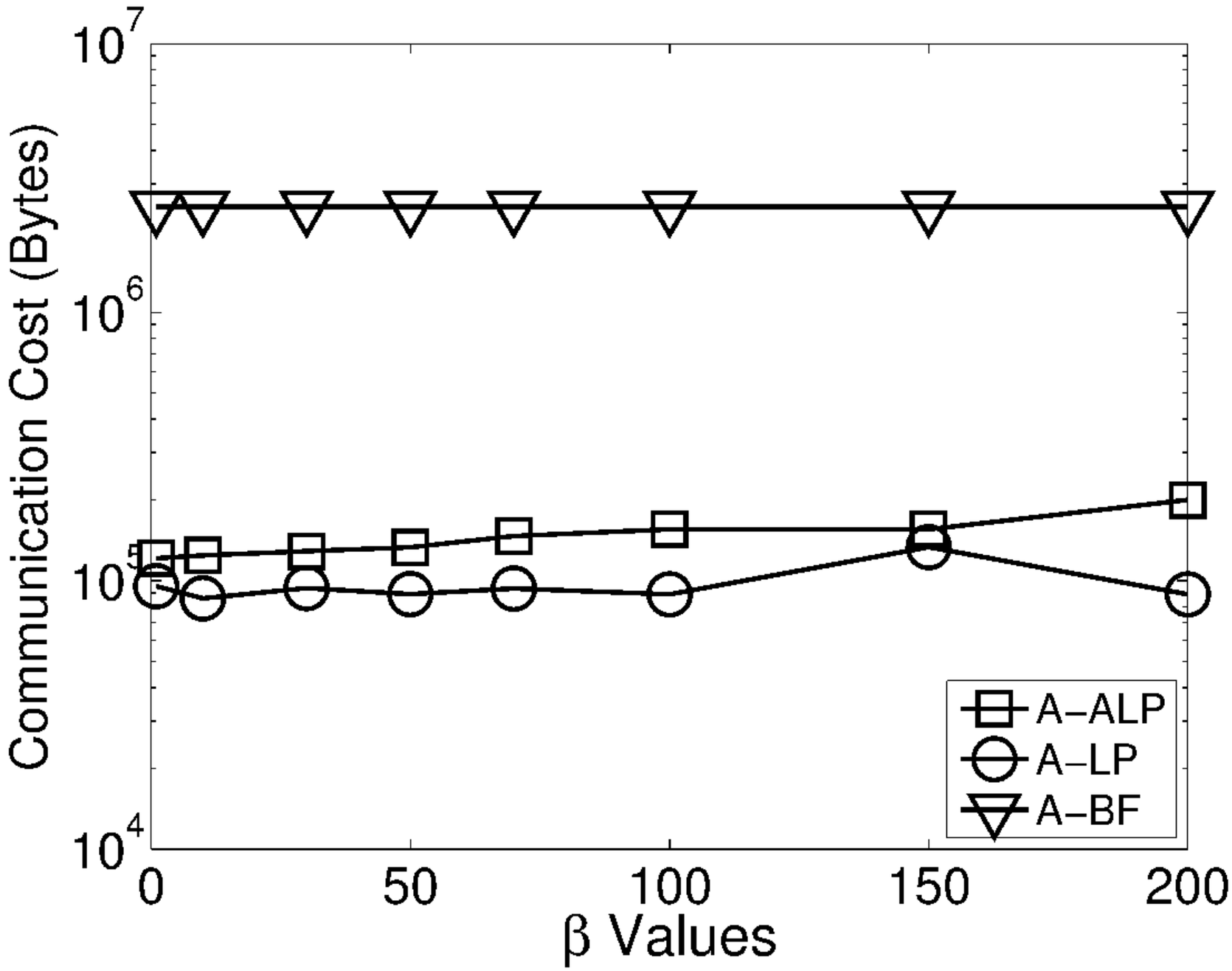
$A - ALP$ Algorithm

Effect of η on Number of Rounds



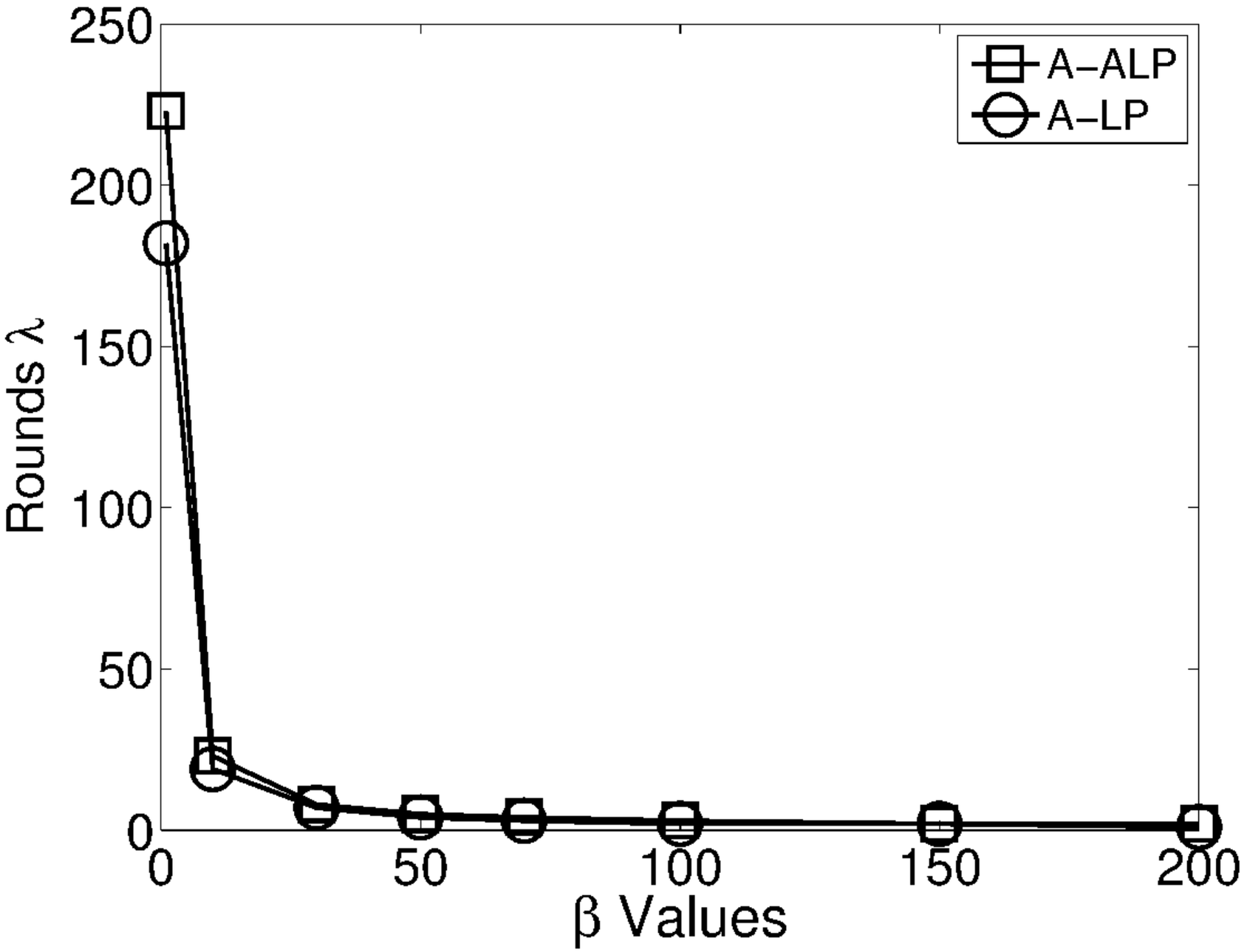
$A - ALP$ Algorithm

Effect of β on Communication Cost



Chlorine Data Set

Effect of β on Number of Rounds



Chlorine Data Set

Conclusions

- We introduced computation and communication efficient algorithms to rank distributed probabilistic data using expected ranks

Conclusions

- We introduced computation and communication efficient algorithms to rank distributed probabilistic data using expected ranks
- For future work we would like to study ranking distributed probabilistic data using other ranking definitions, such as U-kRanks and the Paramterized Ranking Function

Conclusions

- We introduced computation and communication efficient algorithms to rank distributed probabilistic data using expected ranks
- For future work we would like to study ranking distributed probabilistic data using other ranking definitions, such as U-kRanks and the Paramterized Ranking Function
- Both PRF and U-kRanks rely upon the a tuple's **rank distribution** and we believe $A-LP$ and $A-ALP$ could be extended to support these two definitions

Conclusions

- We introduced computation and communication efficient algorithms to rank distributed probabilistic data using expected ranks
- For future work we would like to study ranking distributed probabilistic data using other ranking definitions, such as U-kRanks and the Paramterized Ranking Function
- Both PRF and U-kRanks rely upon the a tuple's **rank distribution** and we believe $A-LP$ and $A-ALP$ could be extended to support these two definitions
- In addition to ranking queries we would like to study other popular queries, such as skyline or nearest neighbor queries, over distributed probabilistic data

The End

THANK YOU

Q and A

- The entire source code is available from a link at

  http:/ww2.cs.fsu.edu/~jestes

Markov Inequality Lower Bound

$$r^{-}(t, D_i) = n_i - \tau \sum_{j=1}^{n_i} \sum_{\ell=1}^{b_{ij}} \frac{p_{i,j,\ell}}{v_{i,j,\ell}} \tag{9}$$

Markov Inequality Lower Bound

$$r^-(t, D_i) = n_i - \tau \sum_{j=1}^{n_i} \sum_{\ell=1}^{b_{ij}} \frac{p_{i,j,\ell}}{v_{i,j,\ell}} \tag{9}$$

- We have these invariants for each site s_i

Markov Inequality Lower Bound

$$r^-(t, D_i) = n_i - \tau \sum_{j=1}^{n_i} \sum_{\ell=1}^{b_{ij}} \frac{p_{i,j,\ell}}{v_{i,j,\ell}} \tag{9}$$

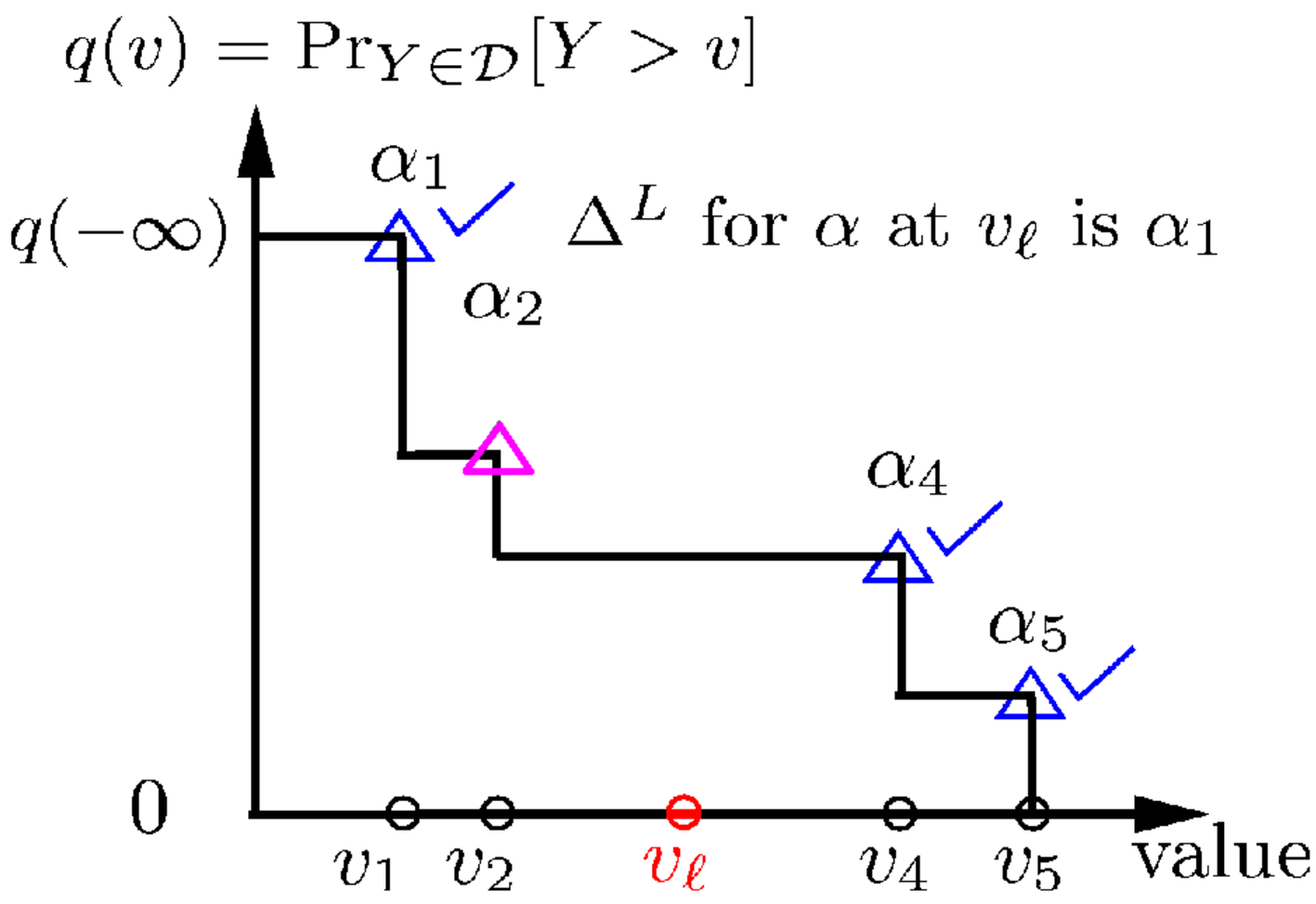
- We have these invariants for each site s_i
- We only have to send these invariants one time to the server and then the server can check the termination condition $r_\lambda^+ \leq r_\lambda^-$ at the end of each round λ

Markov Inequality Lower Bound

$$r^-(t, D_i) = n_i - \tau \sum_{j=1}^{n_i} \sum_{\ell=1}^{b_{ij}} \frac{p_{i,j,\ell}}{v_{i,j,\ell}} \tag{9}$$

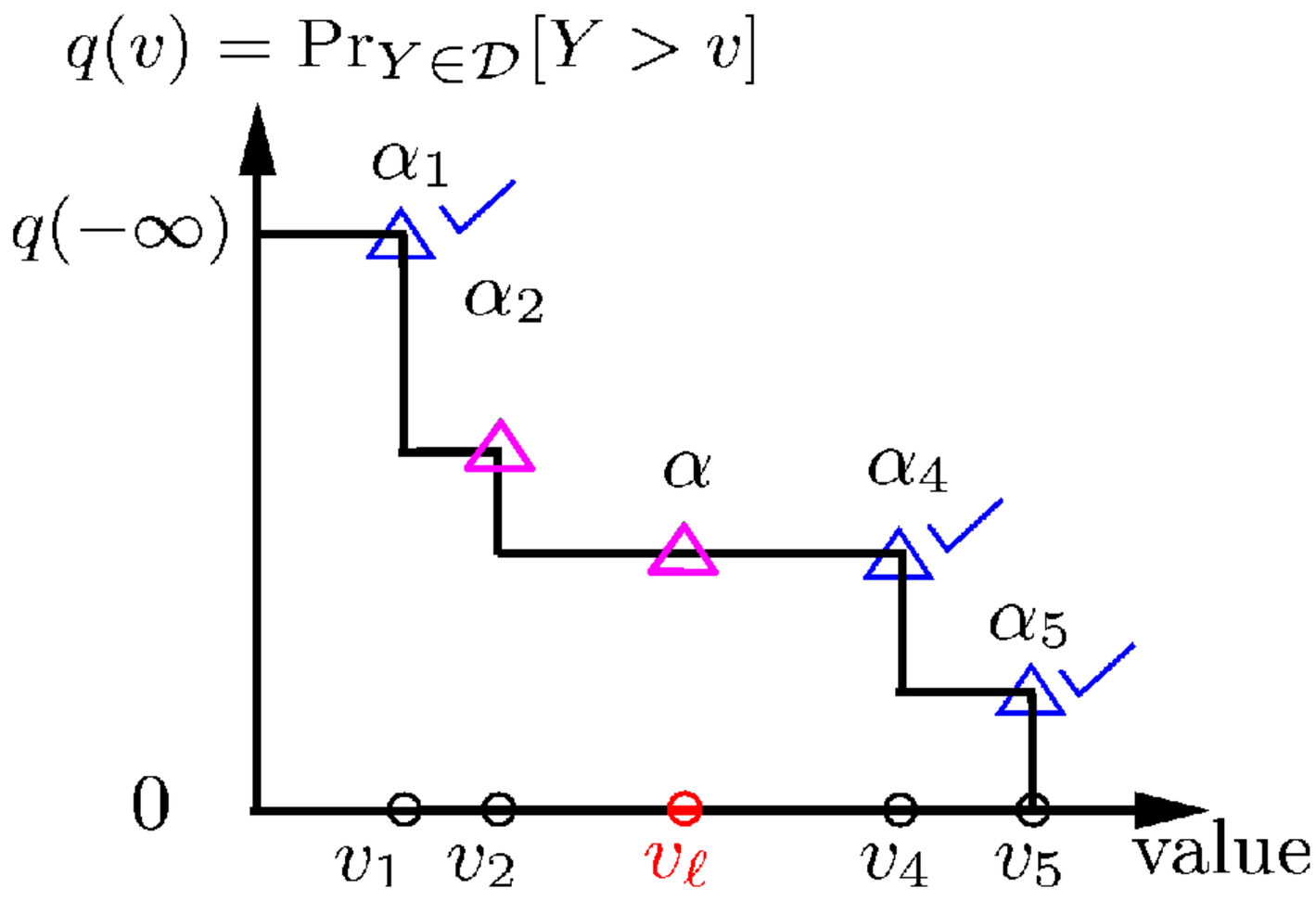
- We have these invariants for each site s_i
- We only have to send these invariants one time to the server and then the server can check the termination condition $r_\lambda^+ \leq r_\lambda^-$ at the end of each round λ
- The Markov Inequality only gives us a loose r_λ^- and this leads directly to our next r_λ^- derivation

Updating the $q^*(v)$'s at the server.... for free



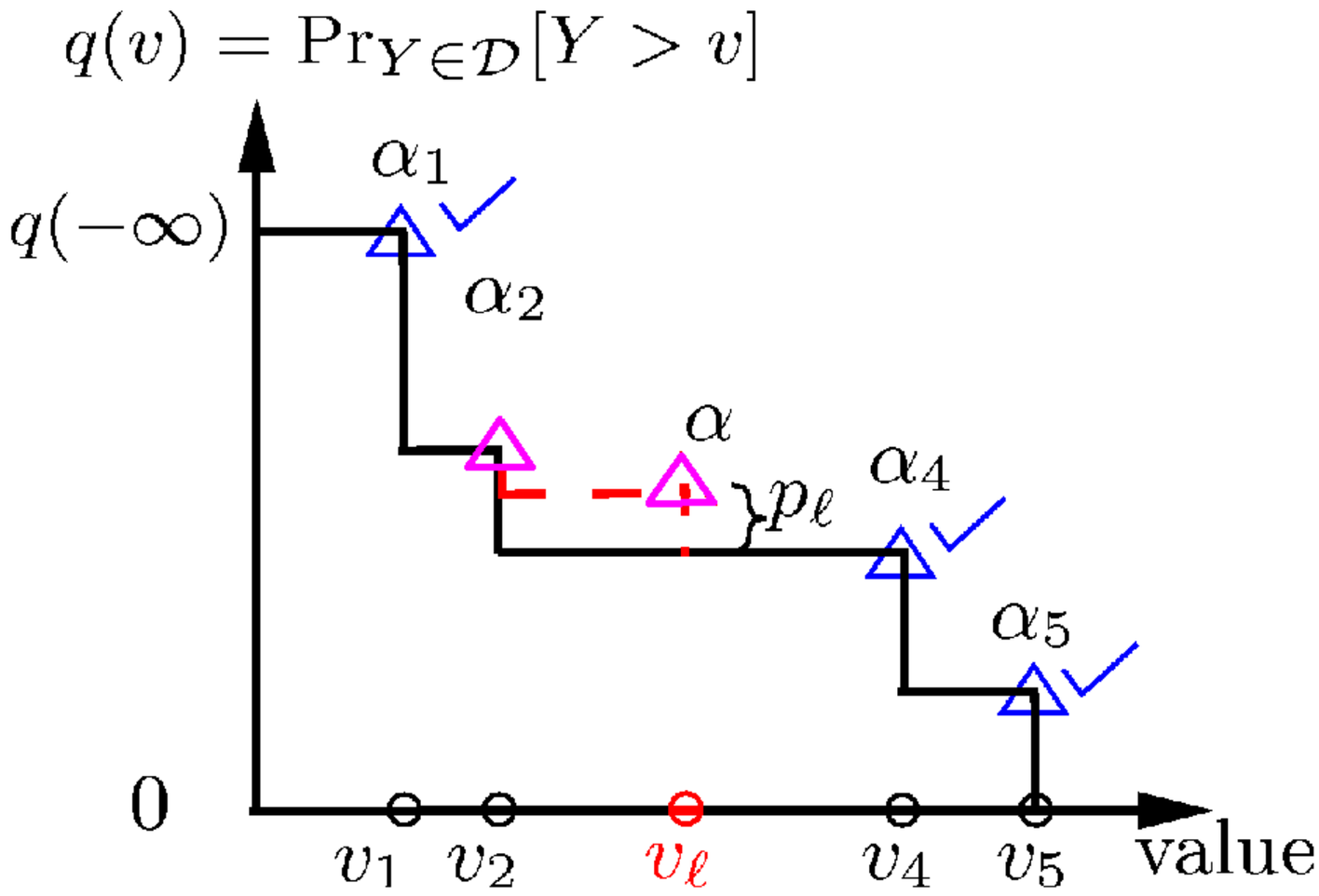
- During some round the server may see a (v_ℓ, p_ℓ) pair from an X s.t. v_ℓ was not an originally sampled upper right corner point from $q(v)$

Updating the $q^*(v)$'s at the server.... for free



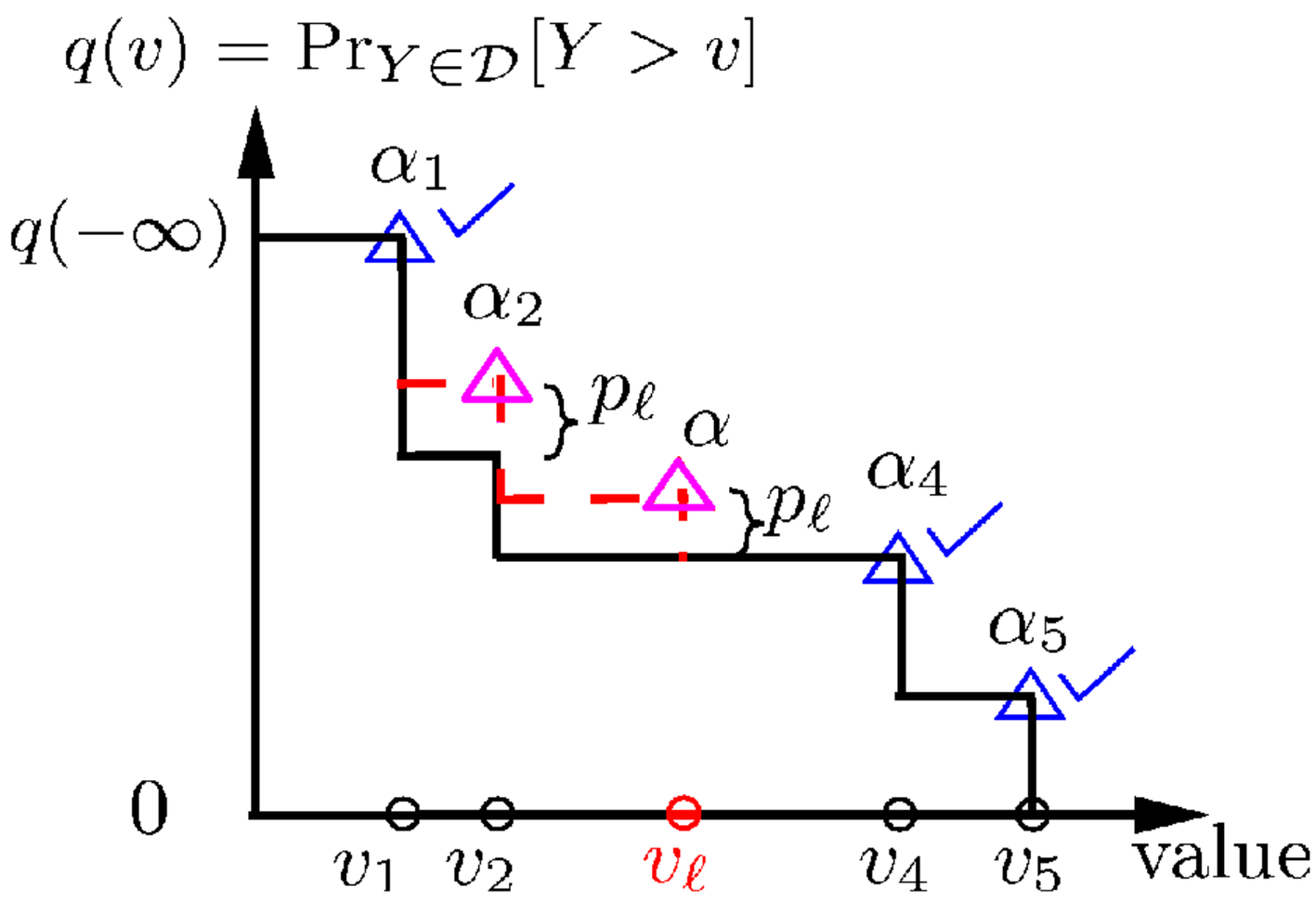
- We create a new α upper right corner point for v_ℓ taking the value of its nearest right neigboring point

Updating the $q^*(v)$'s at the server.... for free



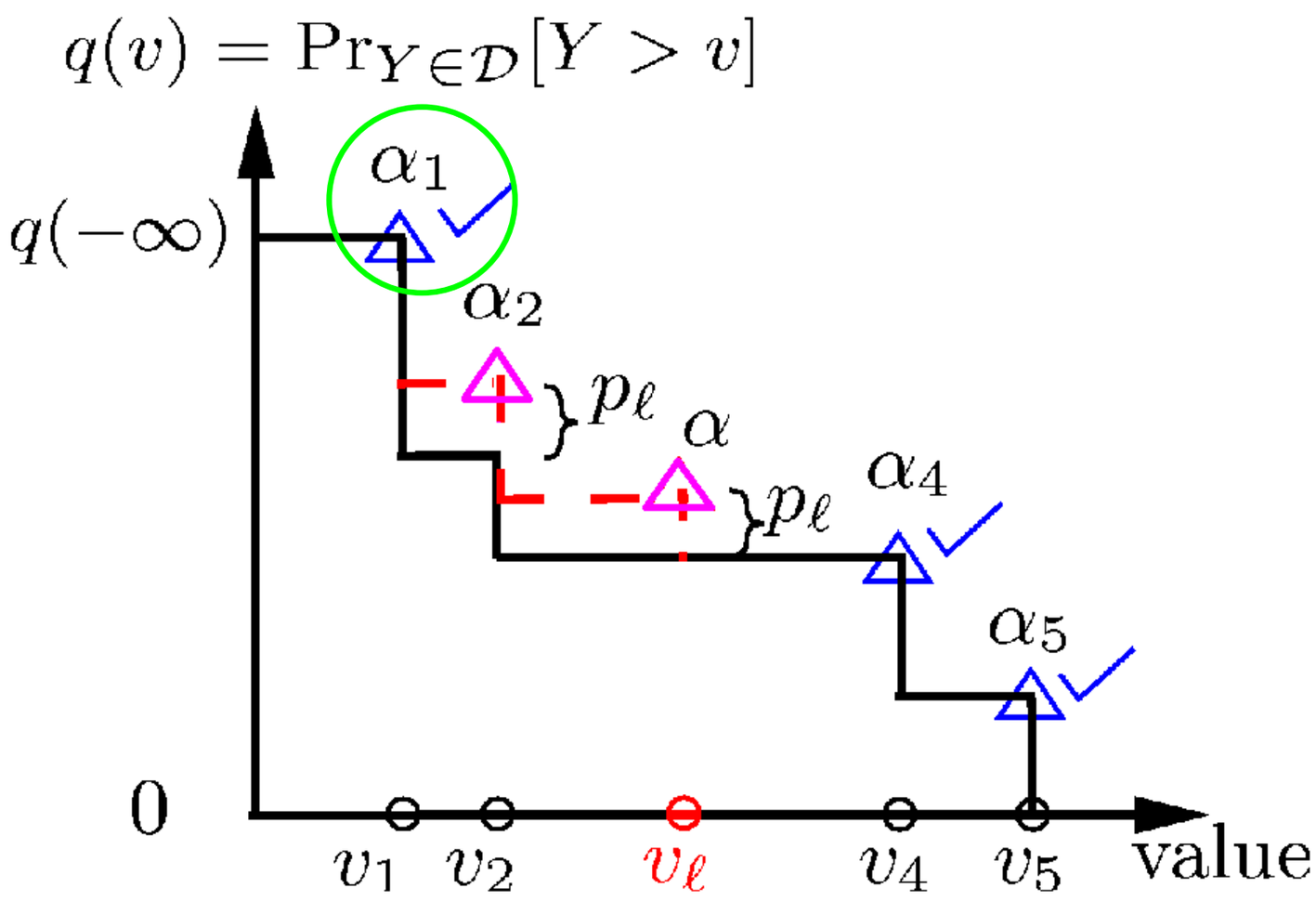
- The new α point is raised by p_ℓ

Updating the $q^*(v)$'s at the server.... for free



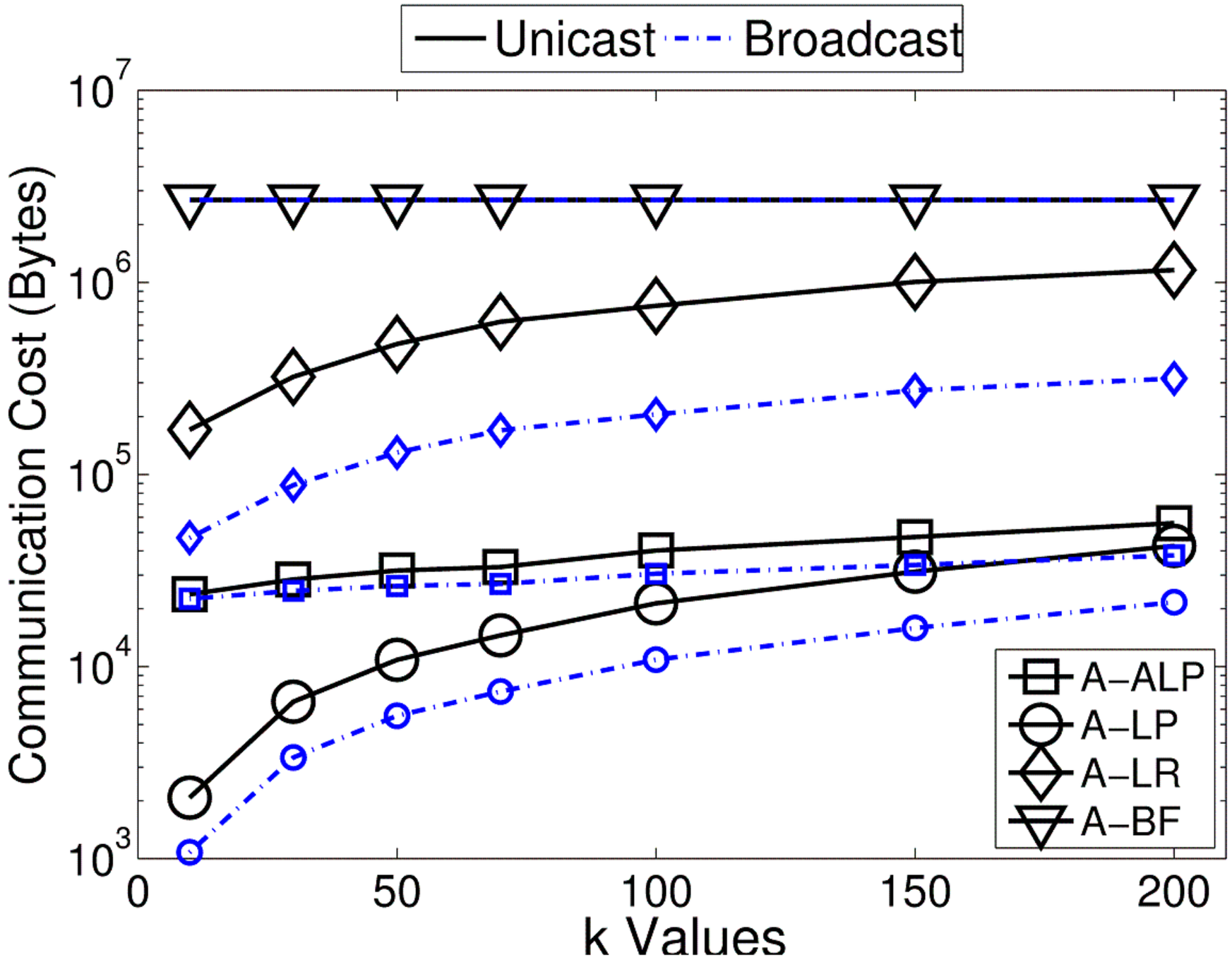
- Any α points to the left of v_ℓ which were not included in the original sampled $q^*(v)$ are also incremented by p_ℓ

Updating the $q^*(v)$'s at the server.... for free



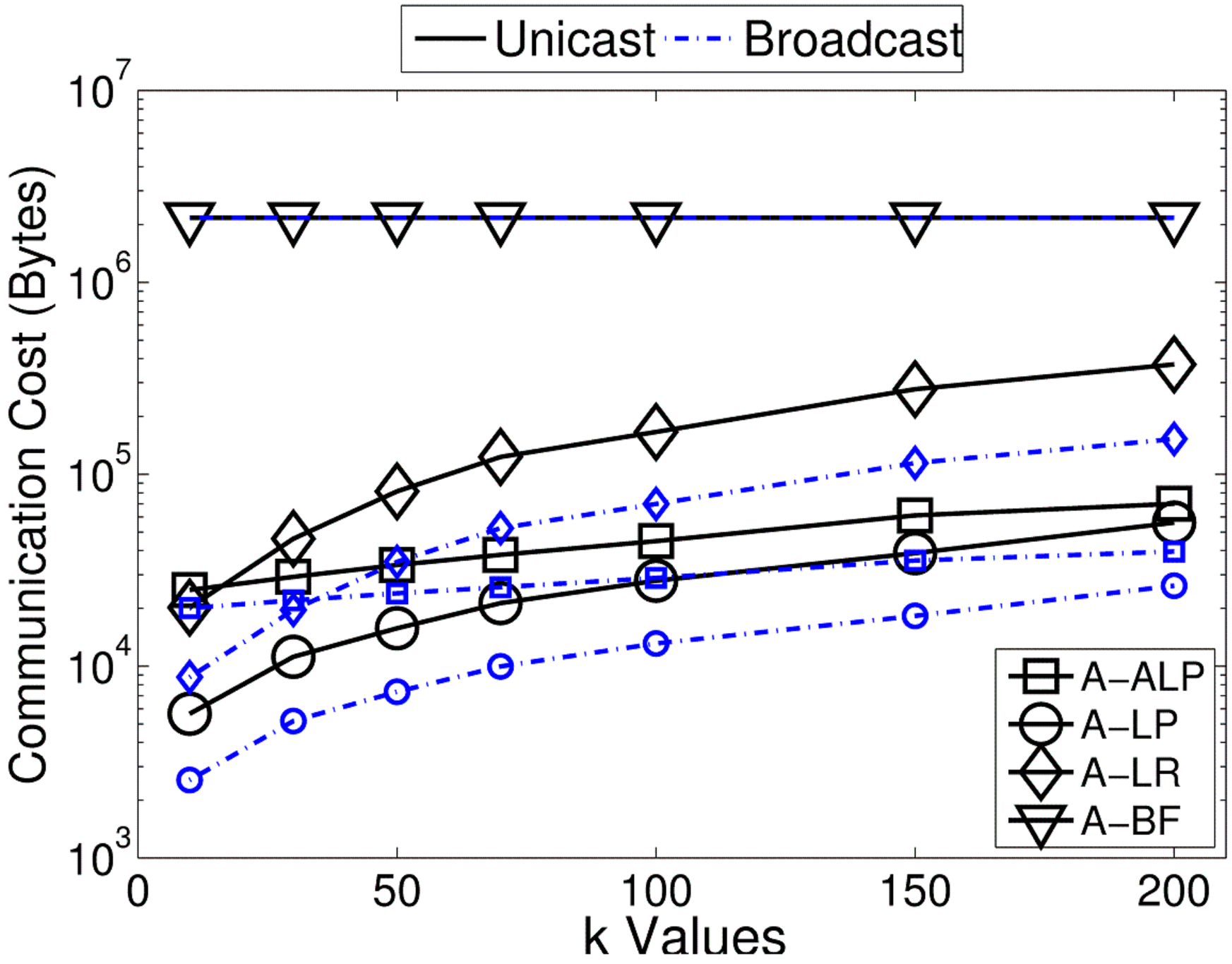
- We stop when we hit the first α point included in the original $q^*(v)$

Communication Cost as k Varies



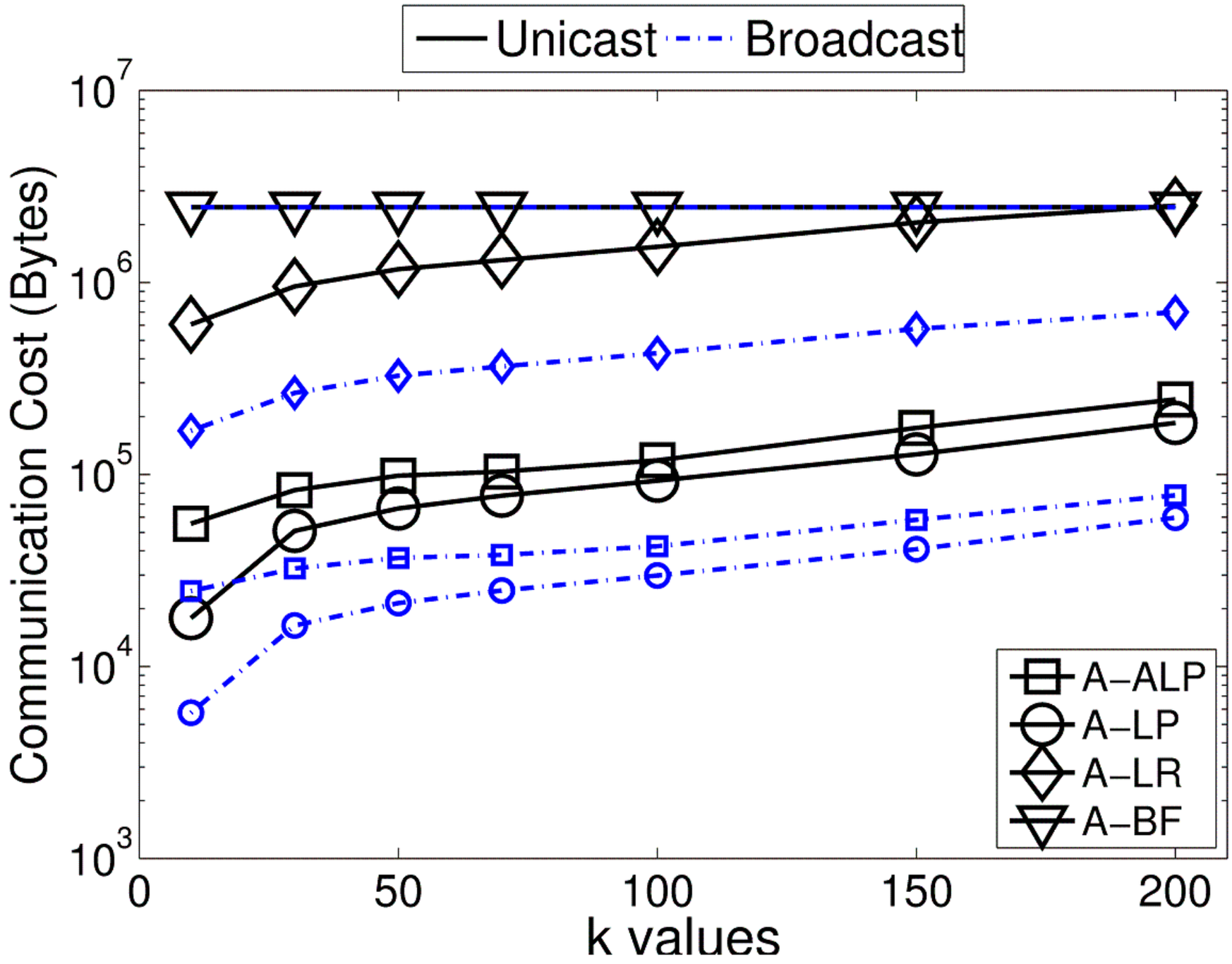
Syntehtic Gaussian

Communication Cost as k Varies



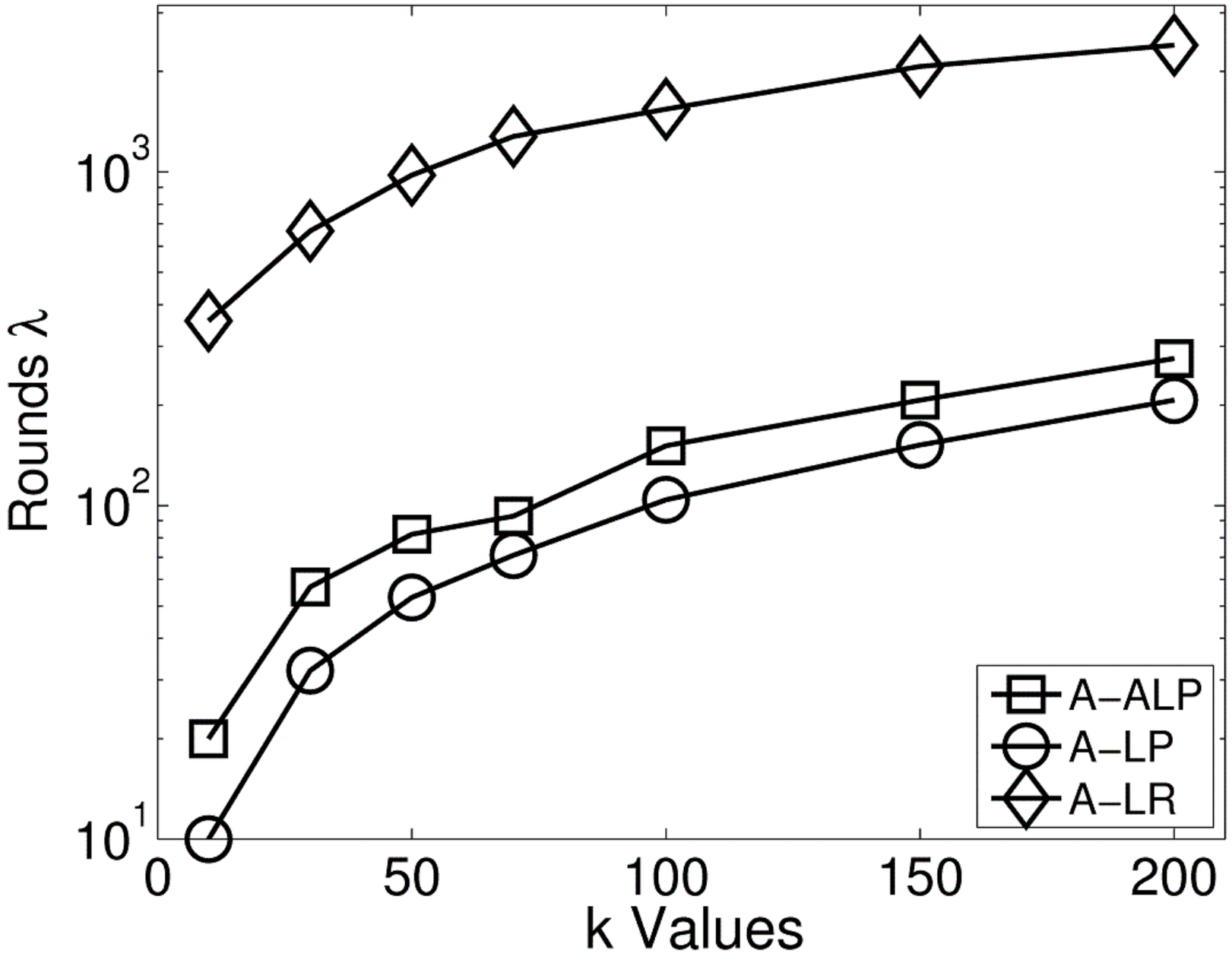
Movie Data Set

Communication Cost as k Varies



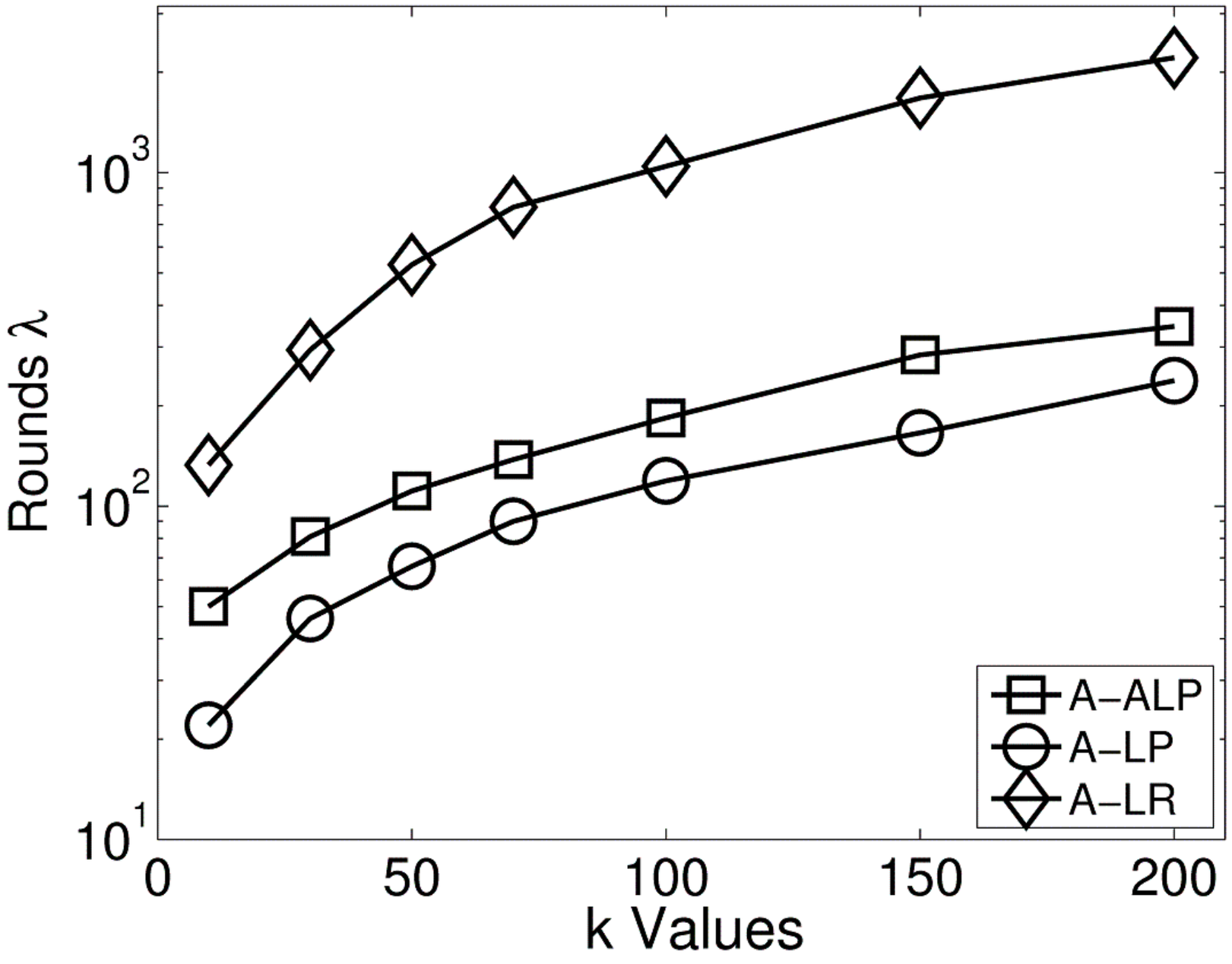
Temperature Data Set

Number of Rounds as k Varies



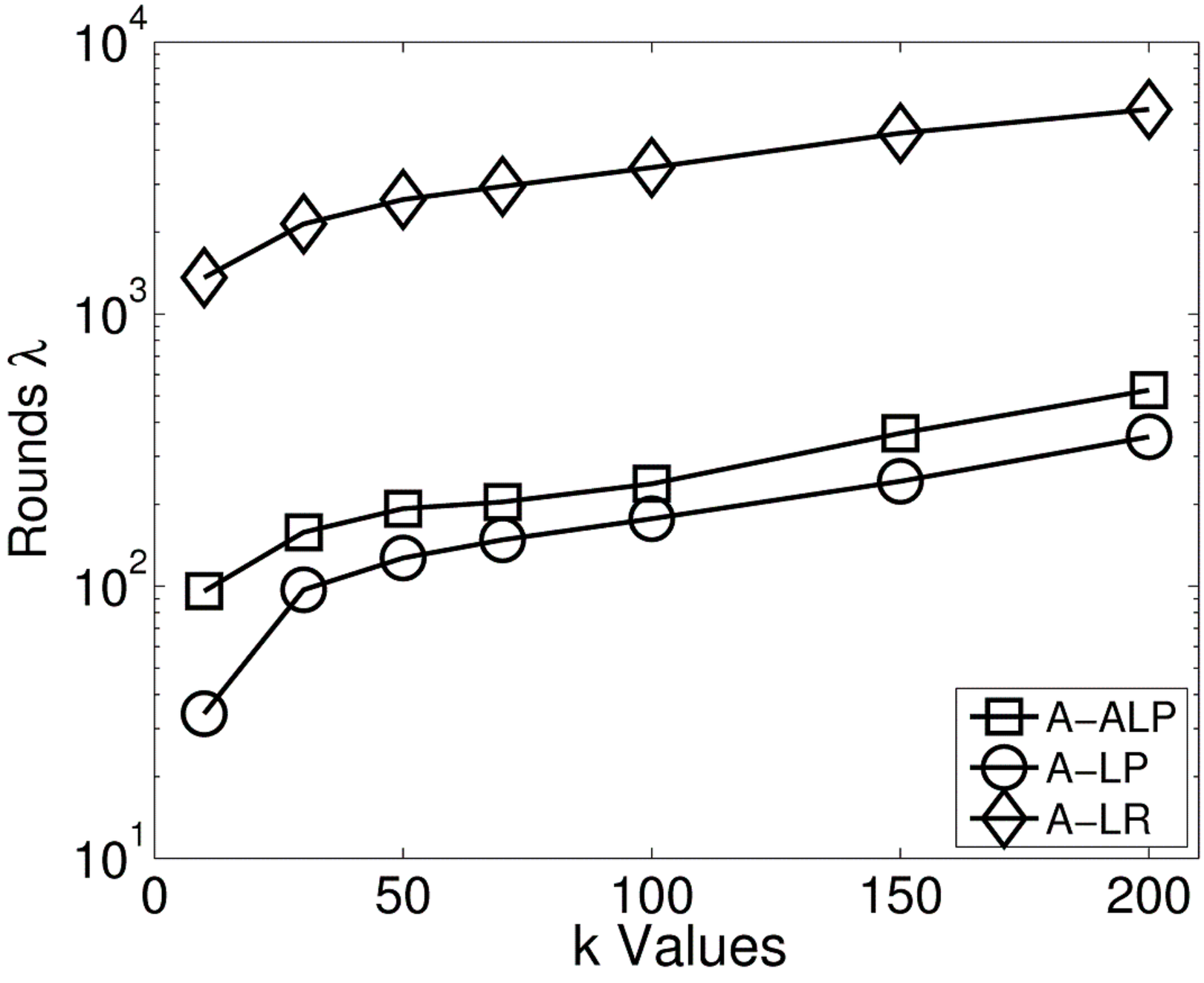
Syntehtic Gaussian

Number of Rounds as k Varies



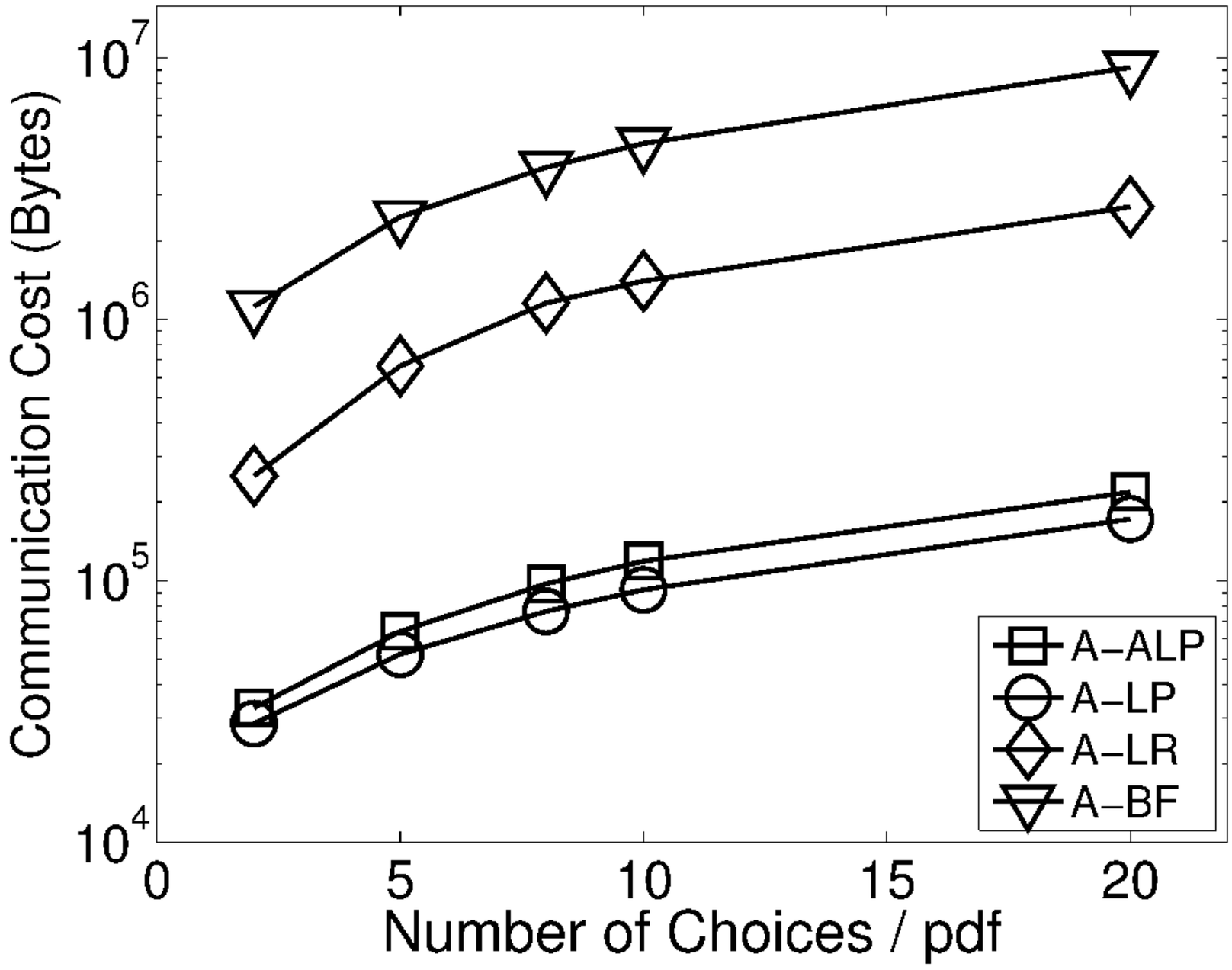
Movie Data Set

Number of Rounds as k Varies



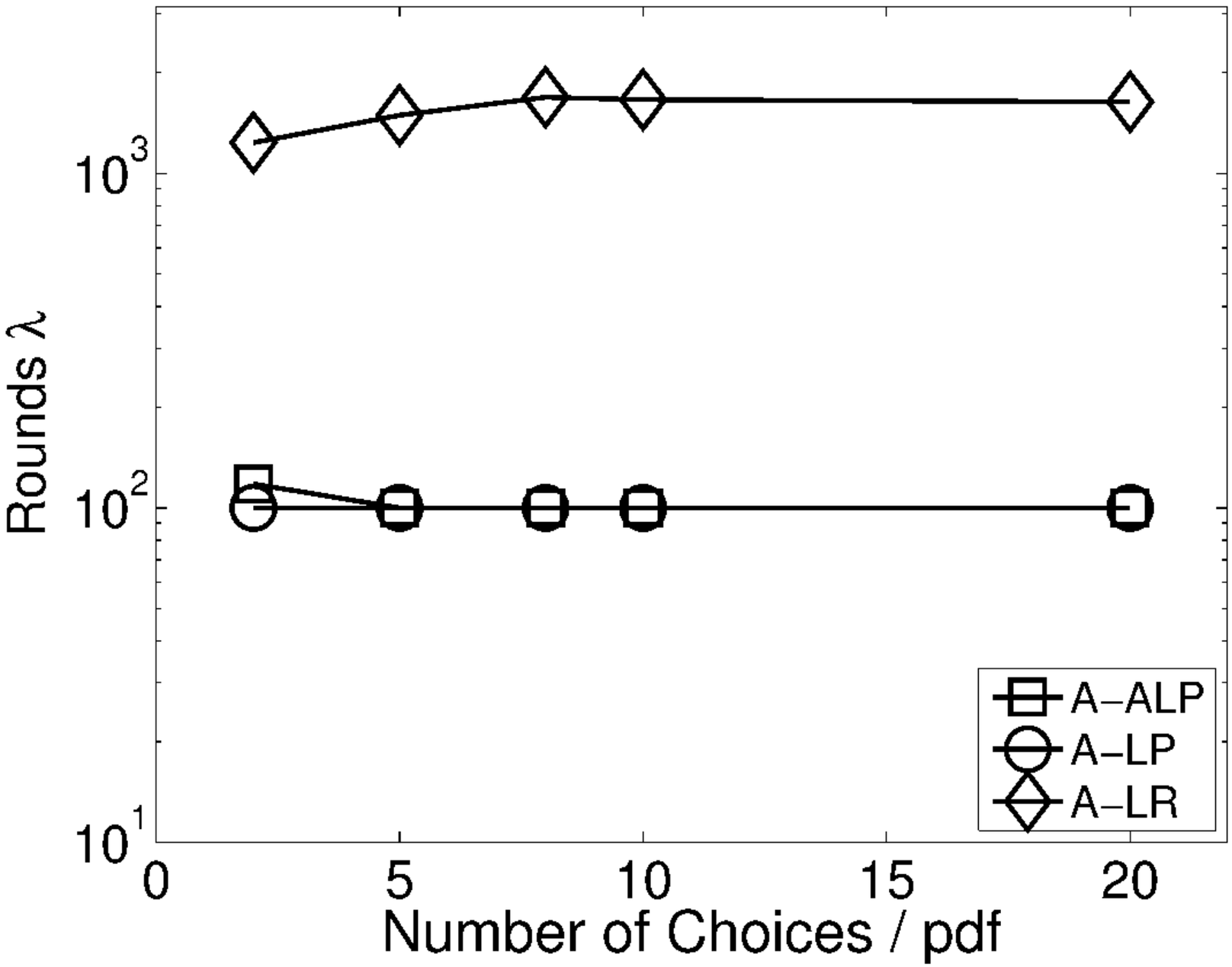
Temperature Data Set

Effect of $|X|$ on Communication Cost



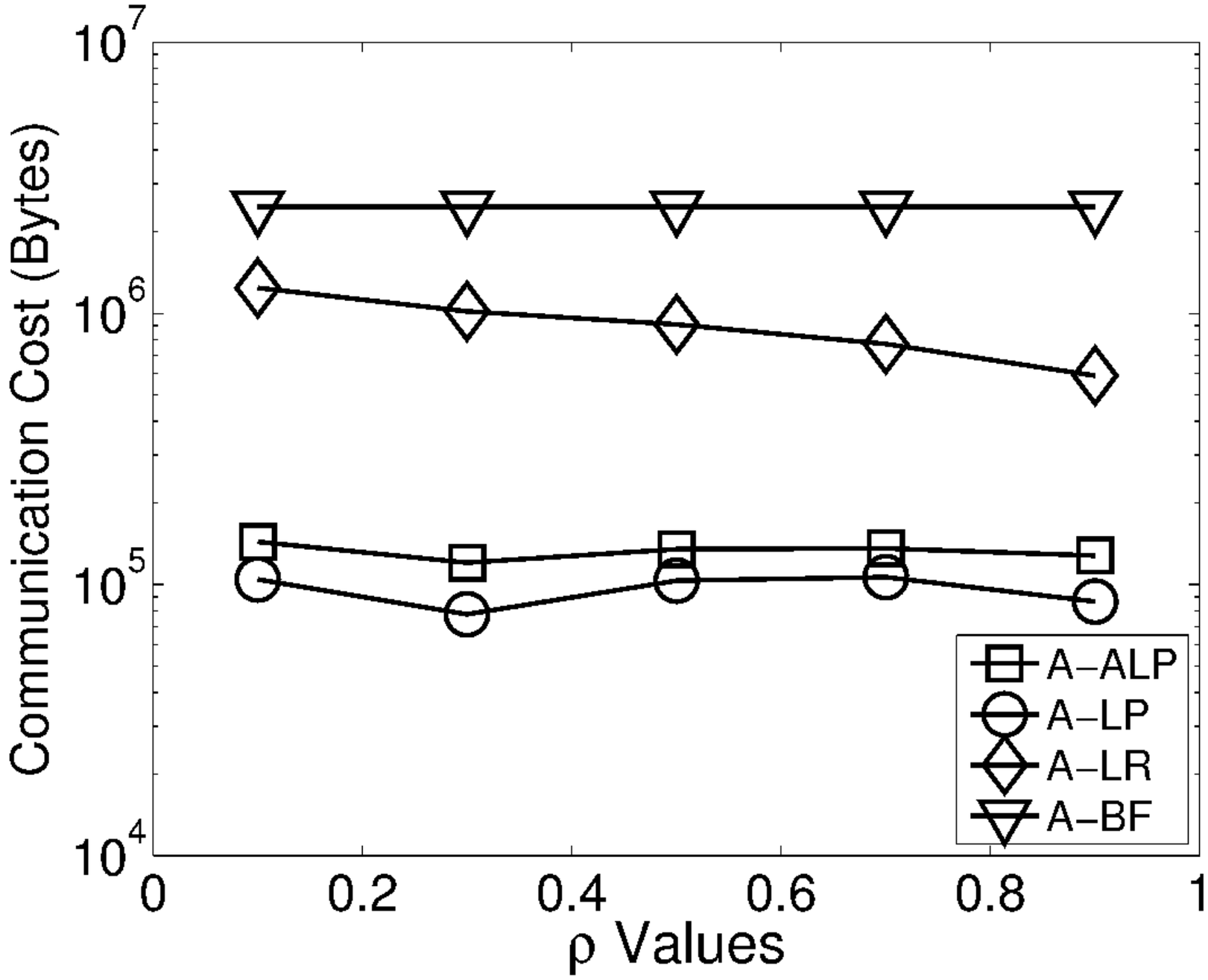
Synthetic Gaussian Data Set

Effect of $|X|$ on Number of Rounds



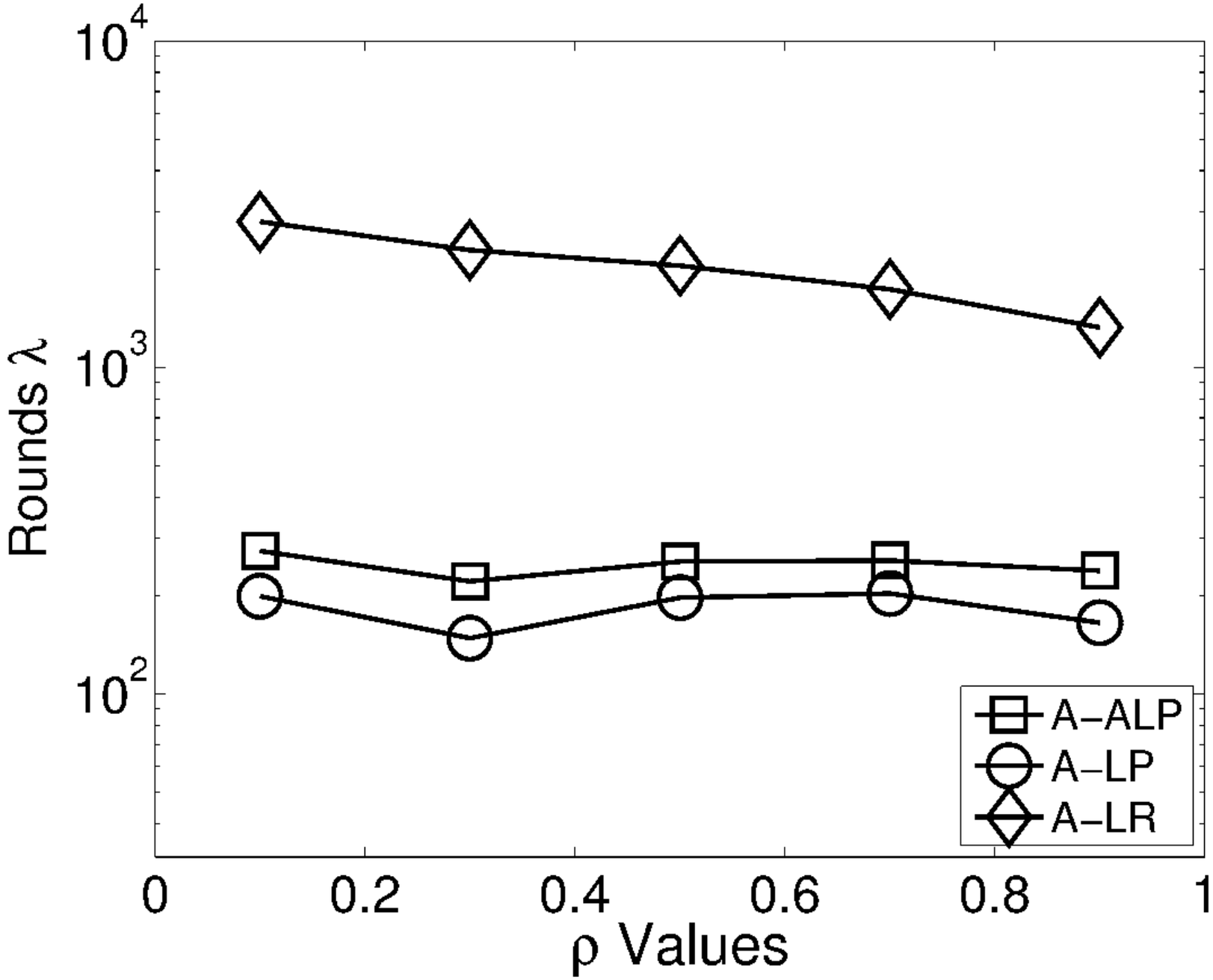
Synthetic Gaussian Data Set

Effect of ρ on Communication Cost



Chlorine Data Set

Effect of ρ on Number of Rounds



Chlorine Data Set