Scalable
Package Queries
in Relational Database Systems

Matteo Brucato
Juan F. Beltran
Azza Abouzied
Alexandra Meliou

Package Queries

- An important class of **combinatorial optimization queries**
- Largely **unsupported** by existing technology
- We make them ***first-class citizens*** in relational databases:
  - The Package Query Language (PaQL)
  - PaQL to Integer Linear Programming (ILP)
- Scalable evaluation through the SketchRefine algorithm

Example: Meal Planning

A dietician wants to build a **Meal Plan** for a patient

Meal Plan

- 3 gluten-free recipes (breakfast, lunch, dinner)
- At least 2 Cal **in total**
- Lowest **total** fat intake

A Meal Plan is a "*Package*"

Meal Plan
- 3 gluten-free recipes (breakfast, lunch, dinner)
- At least 2 Cal
- Lowest total fat intake

Base Constraint (selection predicate)
- **All recipes** in the meal plan are **gluten-free**

Cardinality Constraint
- **3** recipes

Summation Constraint
- **>= 2** Calorie in **total**

Objective Criterion
- **Lowest total** fat intake

Global:
True of the entire set of recipes

Using SQL

Meal Plan

- All gluten-free recipes (selection)
- Exactly 3 recipes (cardinality)
- At least 2 Cal in total (summation)
- Lowest total fat intake (objective)

```
SELECT *
FROM   Recipes R
WHERE  R.gluten = 0
```

Using SQL

Meal Plan

- All gluten-free recipes (selection)
- Exactly 3 recipes (cardinality)
- At least 2 Cal in total (summation)
- Lowest total fat intake (objective)

```
SELECT *
FROM   Recipes R
WHERE  R.gluten = 0
LIMIT  3
```

Using SQL

Meal Plan

- All gluten-free recipes (selection)
- Exactly 3 recipes (cardinality)
- At least 2 Cal in total (summation)
- Lowest total fat intake (objective)

```
SELECT *
FROM   Recipes R1, Recipes R2, Recipes R3
WHERE
  R1.gluten = 0 AND R2.gluten = 0 AND R3.gluten = 0
  AND R1.kcal + R2.kcal + R3.kcal >= 2.0
```

Using SQL

Meal Plan

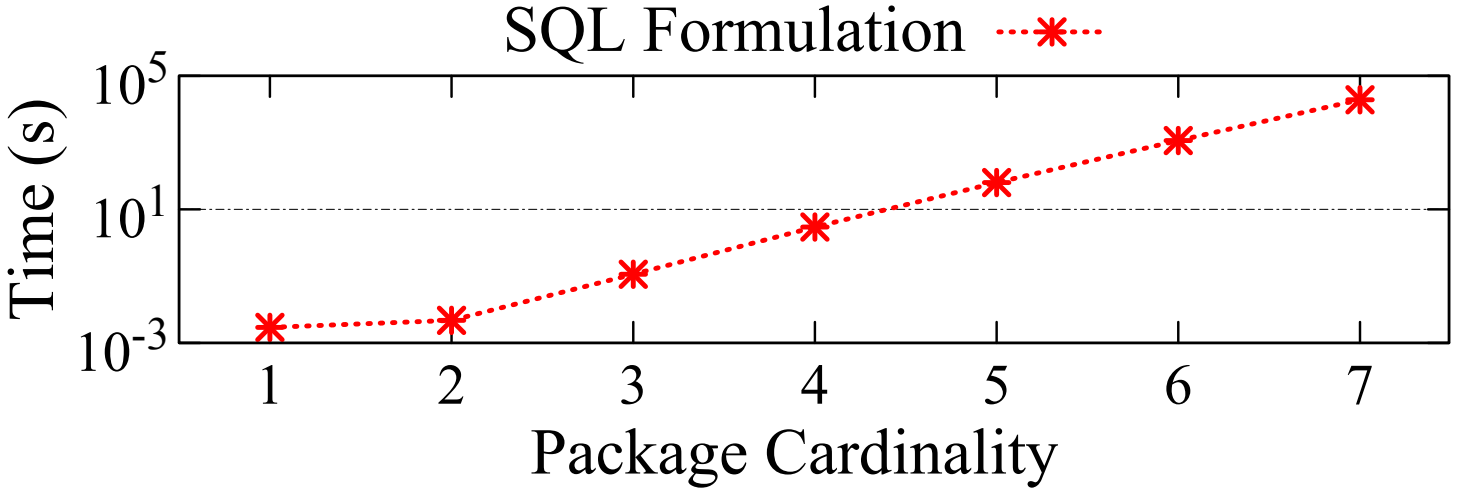
- All gluten-free recipes (selection)
- Exactly 3 recipes (cardinality)
- At least 2 Cal in total (summation)
- Lowest total fat intake (objective)

```
SELECT *
FROM   Recipes R1, Recipes R2, Recipes R3
WHERE
  R1.gluten = 0 AND R2.gluten = 0 AND R3.gluten = 0
  AND R1.kcal + R2.kcal + R3.kcal >= 2.0
ORDER BY   R1.fat + R2.fat + R3.fat
```

Self-joins: Too Expensive!

Meal Plan

- All gluten-free recipes (selection)
- Exactly 3 recipes (cardinality)
- At least 2 Cal in total (summation)
- Lowest total fat intake (objective)



Using SQL

Meal Plan

- All gluten-free recipes (selection)
- Exactly 3 recipes (cardinality)
- At least 2 Cal in total (summation)
- Lowest total fat intake (objective)

Self-joins are able to express **cardinality** and **linear** constraints.
But what if there is **no cardinality constraints**?...

Using SQL

Meal Plan

- All gluten-free recipes (selection)
- Exactly 3 recipes (cardinality)
- At least 2 Cal in total (summation)
- Lowest total fat intake (objective)

How many self-joins?...

It needs SQL's full recursive power to build all candidate subsets!

Recap: Why is SQL insufficient?

1. With known cardinality, **self-joins** can express package queries, but they are **too expensive to compute**
2. Without known cardinality, self-joins are **not powerful enough**
3. The self-join solution **loses in declarativeness**

Packages as first-class citizens in database systems

- **Packages** are **not unique** to each application
- Nor are the **algorithms** for constructing them
- The **data** typically resides in a database

We introduce:

- A declarative package query language: **PaQL**
- PaQL to ILP **translation** method
- **Scalable** PaQL evaluation method

PaQL: The Package Query Language

Meal Plan
- All gluten-free recipes (selection)
- Exactly 3 recipes (cardinality)
- At least 2 Cal in total (summation)
- Lowest total fat intake (objective)

```
SELECT PACKAGE(R)
  FROM Recipes R
 WHERE R.gluten=0
SUCH THAT COUNT(*)=3 AND
          SUM(kcal)>=2
MINIMIZE SUM(fat)
```

DIRECT: PaQL to ILP Translation

- An ILP problem consists of:
  - A set of **integer variables**
  - A set of **linear constraints** over the variables
  - A **linear objective function** over the variables
- An ILP solution consists of:
  - An **assignment** to each of the integer variables

PaQL to ILP: Variables

- Integer Variables: one for each tuple

Recipes

kcal	fat
26	2.4
120	8.1
13	0.1
54	2.1

t1 → **x1**

t2 → **x2**

… **…**

tn → **xn**

How many times is **t1** repeated in the result package?

$x1 \in \mathbb{Z}, x1 \geq 0$

PaQL to ILP: Global Conditions

- **SUCH THAT**
  - COUNT(*) = 3

    $\Sigma_i \, x_i = 3$
  - SUM(kcal) >= 2

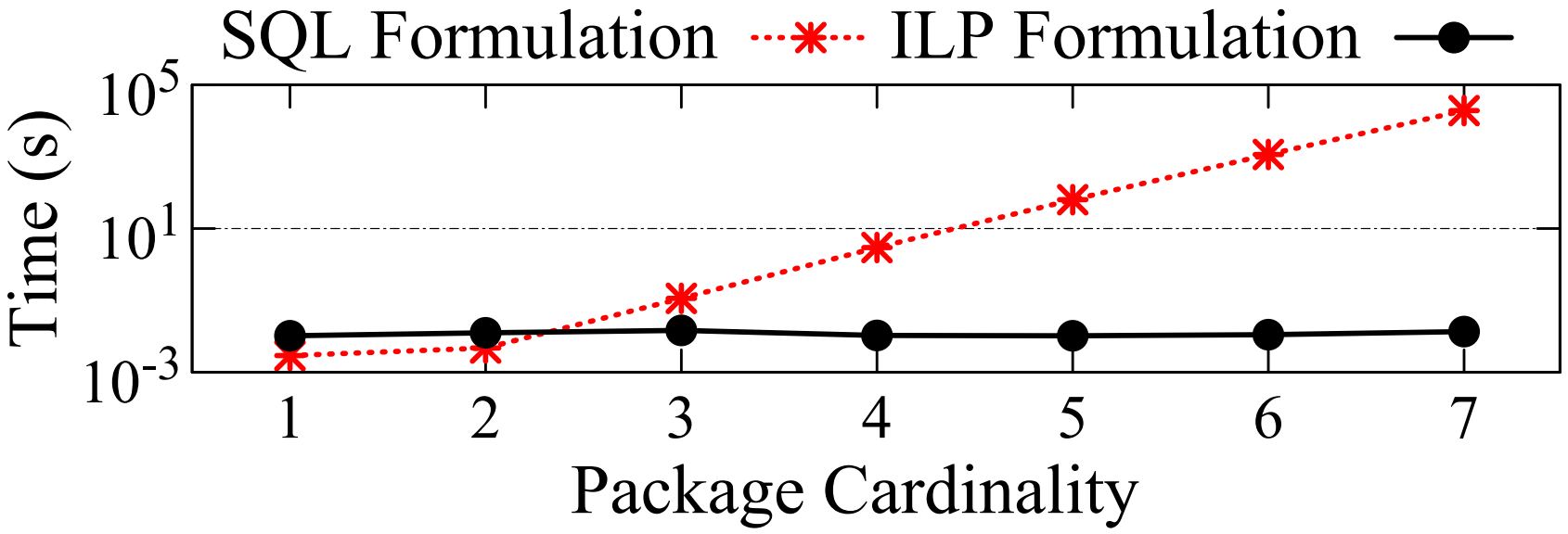
    $\Sigma_i \, (t_i.\text{kcal} \cdot x_i) \geq 2$

- Objective Criterion
  - MINIMIZE SUM(fat)

    $\text{minimize } \Sigma_i \, (t_i.\text{fat} \cdot x_i)$

DIRECT: Query Evaluation

1. Apply base predicate (selections)
2. Formulate ILP
3. Solve ILP



Drawbacks of DIRECT

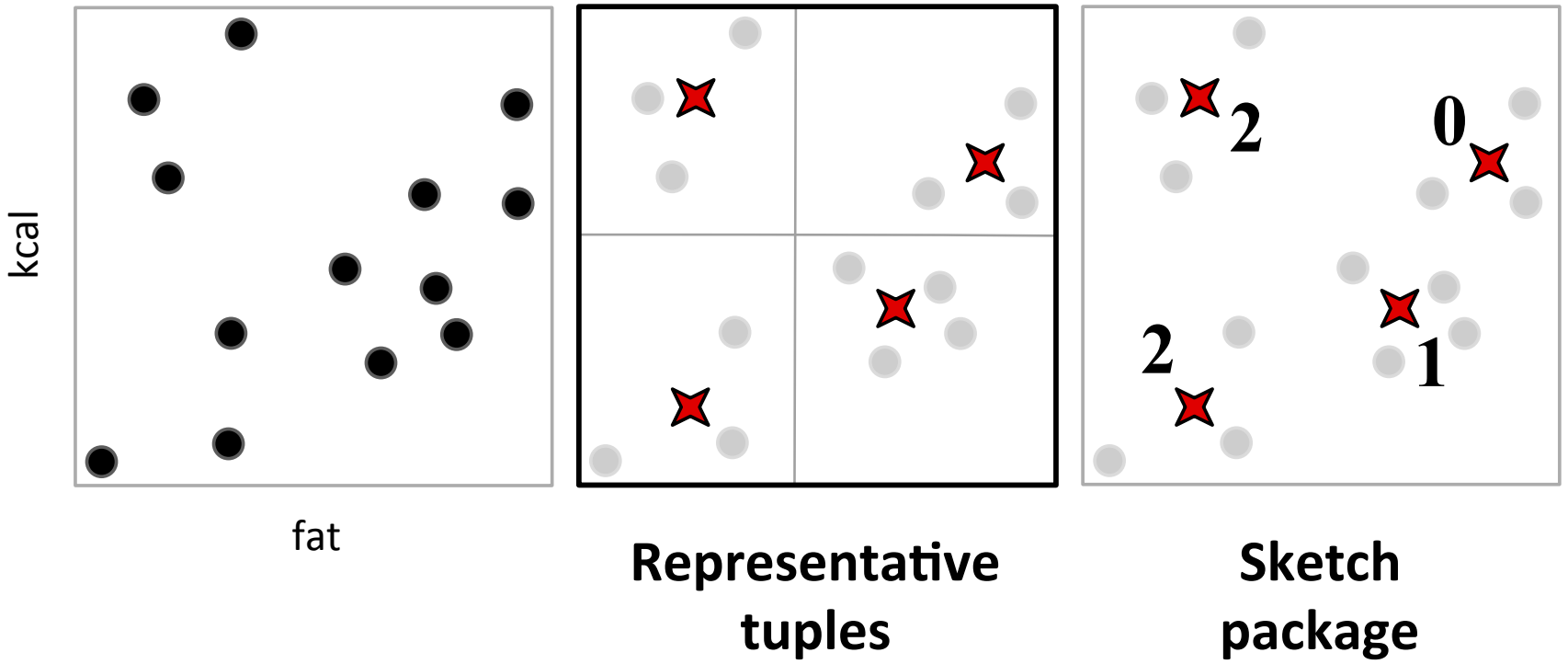
- Only applicable if data **fits** entirely in main memory
- It **may fail** due to the **complexity** of the ILP problem

SketchRefine: Scalable Evaluation

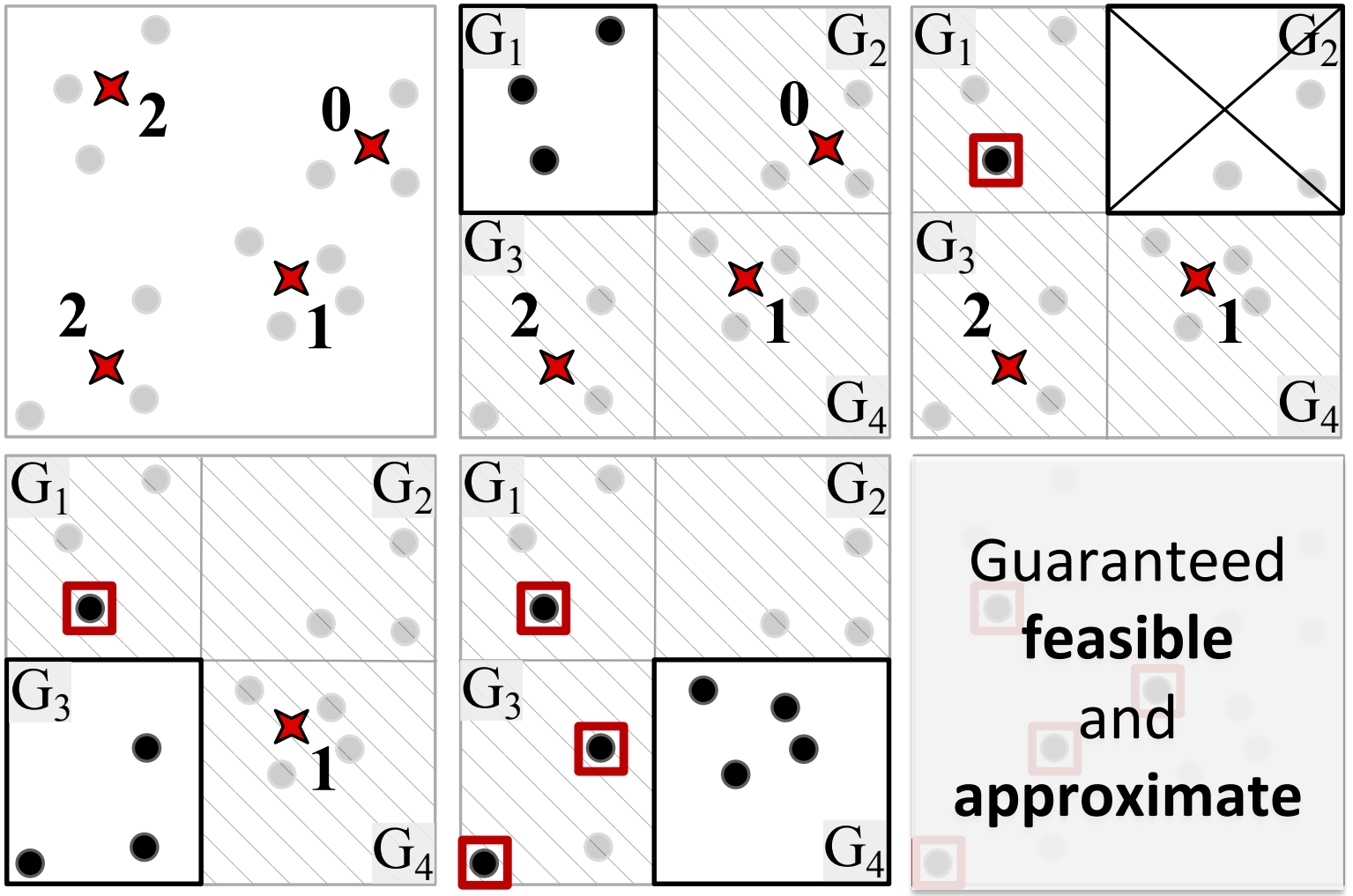
- Partition Data (offline)
  - Into groups of similar tuples
  - Elect a "representative" tuple for each group
- Sketch
  - an initial package from the representatives
- Refine
  - the initial package using real tuples

Returns approximate, feasible, package

Partition and SKETCH



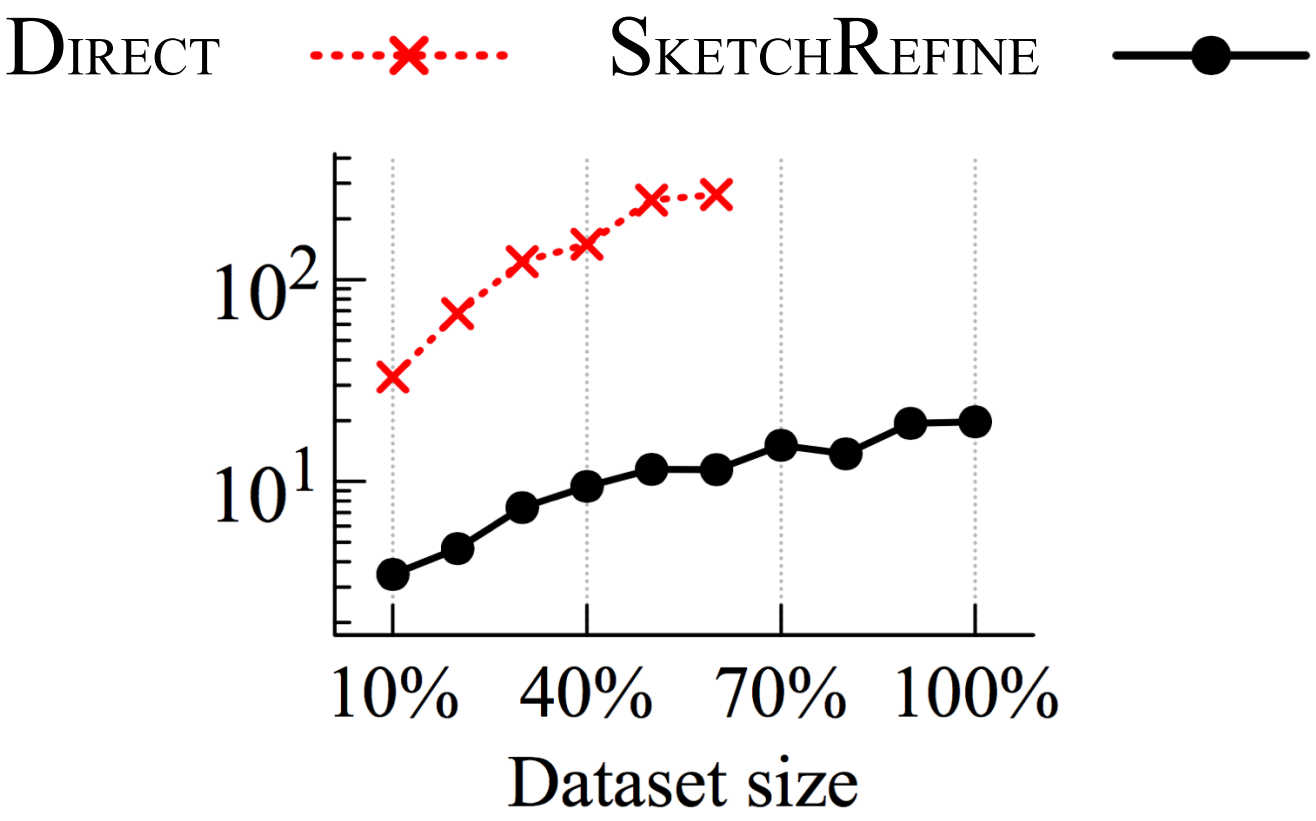
REFINE


2
0
2
1
G1
G2
0
G3
2
1
G4
G1
G2
G3
1
G4
G1
G2
G3
G4
Guaranteed
feasible
and
approximate

Approximation Guarantees

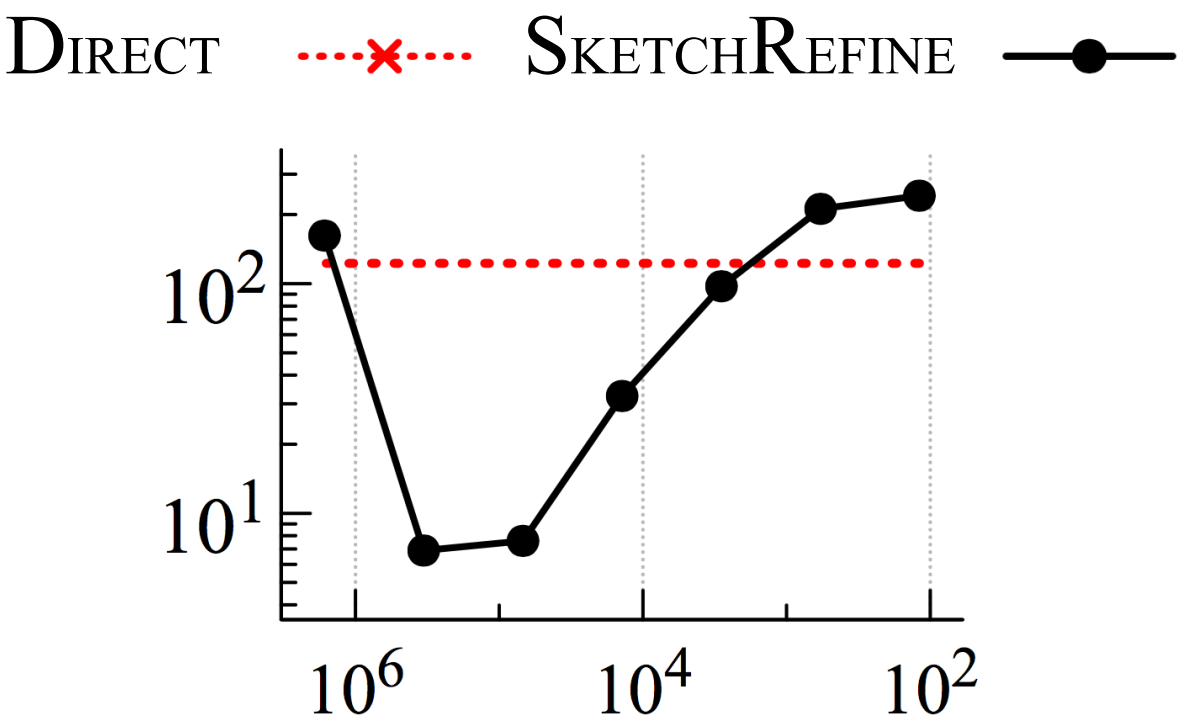
- SketchRefine is a $(1 \pm \varepsilon)^6$-approximation with respect to Direct
- ε limits the partitions radius
- The partition size (number of tuples per partition) does not affect the approximation guarantee

Scalability



Approximation Ratio:
Mean: 1.01, Median: 1.00

Partition Size: Performance Impact



Approximation Ratio:
Mean: 1.01, Median: 1.00

Thank You!

Matteo Brucato

matteo@cs.umass.edu